A Statistical Learning Framework for Data Mining of Large-Scale Systems: Algorithms, Implementation, and Applications

by

Ching-Huei Tsou

M.Eng., Civil and Environmental Engineering
Massachusetts Institute of Technology, 2002

S.M., Civil Engineering
National Taiwan University, 1997

Submitted to the Department of Civil and Environmental Engineering
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

at the
Massachusetts Institute of Technology
February 2007

Signature of Author: _____
Department of Civil and Environmental Engineering
December 18, 2006

Certified by: _____
John R. Williams
Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by: _____
Andrew Whittle
Chairman, Department Committee on Graduate Students

A Statistical Learning Framework for Data Mining of Large-Scale Systems: Algorithms, Implementation, and Applications

by

Ching-Huei Tsou

Submitted to the Department of Civil and Environmental Engineering
On January 10, 2007 in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy in the Field of
Information Technology

ABSTRACT

A machine learning framework is presented that supports data mining and statistical modeling of systems that are monitored by large-scale sensor networks. The proposed algorithm is novel in that it takes both observations and domain knowledge into consideration and provides a mechanism that combines analytical modeling and inductive learning. An efficient solver is presented that allow the algorithm to solve large-scale problems efficiently. The solver uses a randomized kernel that incorporates domain knowledge into support vector machine learning. It also takes advantage of the sparseness of support vectors and this allows for parallelization and online training to further speed-up of the computation. The solver can be integrated into existing systems, embedded into databases, or exposed as a web service.

Understanding the data generated by large-scale system presents several problems. First, statistical modeling approaches may either under-fit or over-fit the data and are sensitive to data quality. Second, learning is a computational extensive process and often becomes intractable when the sample size exceeds several thousands. Third, learning algorithms need to be tuned to the specific problem in most engineering and business fields. Last but not least, a flexible learning framework is also not available. This work addresses these problems by presenting a methodology that combines machine learning with domain knowledge, and an efficient framework that supports the algorithm. Benchmark and practical engineering problems are used to validate the methodology.

Thesis Supervisor: John R. Williams
Title: Professor of Civil and Environmental Engineering

ACKNOWLEDGMENTS

Professor John. R. Williams for his insights, patience, and encouragements. His passionate curiosity and eager for learning has always been a great inspiration.

Professor Sanjay Sarma, Professor Jerome Connor, and Professor Brian Subirana for the knowledge, guidance, and advices.

My father, who is always my hero and biggest role model. My mother, for her endless love. It is because of her I am who I am today.

My wife, who is always there for me. For being my cheer squad, and bring in our precious bundle of joy with perfect timing.

My friends. Everyone in IESL Lab and Auto-ID Lab. For making this a fun journey in addition to hard working.

Engineering Systems Division, the Shell Group, Wal-Mart Stores, the Gillette Company, Procter & Gamble, and EPC Global for providing funding for my research.

TABLE OF CONTENTS

Chapter 1 Introduction

An overview of the problems and the proposed approaches is given in this chapter.

1.1 Motivations

1.1.1 Large Volume of Data

Progress in sensor technology give us the opportunity to monitor various systems in detail, for example, the acceleration responses of a high-rise building, down-hole measurements of an oil well, shipping of merchandizes throughout a supply chain, and replenishing process in a retail store, just to name a few. Together with the rapid development in digital data acquisition and storage technology, those advances have lead to tremendous amount of data and huge databases. This phenomenon is happening in a variety of fields spread from science, engineering, to business. However, uninterrupted data are simply a collection of observations representing limited aspects of a particular event happened in the past within a particular context. How to turn data into information, and use the information to derive actions, have become the new challenges.

1.1.2 A Multidisciplinary Problem

Given the sheer volume of the data, any nontrivial analysis process is obviously unattainable for the limited human capabilities. The science of extracting useful information from large data sets is known as data mining. It is a relatively new research area that draws algorithms from statistics, machine learning, pattern recognition, and other fields. When applying data mining techniques in addressing a specific problem, the domain knowledge of the system of interest is also crucial to the analysis. A reliable solution cannot be achieved without a system-level approach that considers as many aspect of the problem as possible, because many practical problems we are facing today are naturally multidisciplinary. In practice, data analysis is often done by the domain experts with out-of-box data mining algorithms, which are not turned for the problem and hard to customize.

1.1.2 Needs from Real-World Problems

The second motivation comes from the demands of modeling and simulation in real-world problems. The need of turning data into actions appears universally in engineering and business worlds. For example, a central part of the smart-well project running by Shell Oil Company [1, 2], is to improve hydrocarbon production though the use of measure, model, and control, especially in a closed-loop manner as illustrated in figure 1.1. The closed-loop control is a general approach from the classical control theory. Conceptually, the outputs of the system is measured and then feed to a model that approximate the real system. The model deciding the best control strategy based on the processed measurements iteratively, and the result is used as input to the system and which closes the loop. It is obvious that it is crucial to develop a robust mathematical model of the system, and is also the major part of the smart-well project. Traditionally, a mathematically model is created based on domain knowledge about the system. This analytical approach usually leads to a high order physical model that may overly complex or not exist for some less known systems. A probabilistic-based proximate model is a much desirable alternative in the sense it is established mostly on measurements and usually has lower order and are computational more efficient.

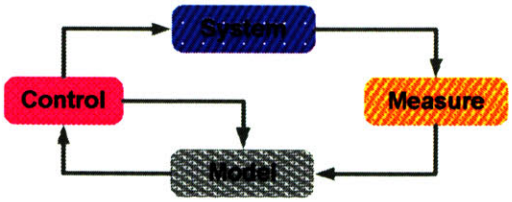


Figure 1.1 Measure-Model-Control Loop

As another example, Wal-Mart, the world's largest retailer, started introducing electronic product code (EPC) and radio frequency identification (RFID) technology into its supply chain in 2004, and issued a mandate to its suppliers to tag their merchandizes [3]. Wal-Mart begins receiving cases and pallets of product with EPC tags in pilot stores since April 30, 2004, and starts related business process change since a year after. Various data are collected during the initiative, such as the EPC reads from suppliers' factories to Wal-Mart's stores, daily point-of-sales data, and out-of-stock data in both pilot and control stores. To quickly learn the return on investment, bottleneck in process change, and any possible policy resistances, studies from both Wal-Mart and its suppliers had been conducted soon after the initiative. Each research group adopted different modeling approaches, most of them overly simplified due to the time constraint. Inconsistent results have been reported from each group although similar data sets were used.

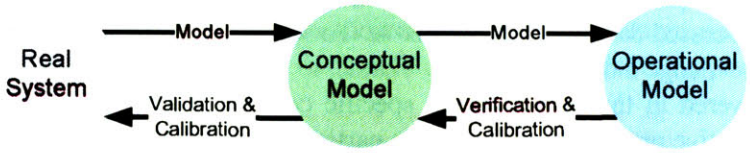


Figure 1.2 Iterative Modeling Process

To create a realistic model, the modeling process needs to be done in an iterative manner, as shown in figure 1.2. A modeling process is always a competition between two conflicting goals: realism and simplicity. On one hand, a model needs to be a reasonably close approximation of the real system; while on the other hand, a model should not be overly complex and often needs to be delivered in a timely fashion.

1.1.3 The Future of Machine Learning

The third motivation comes from the improvement of the state of the art of machine learning techniques. Machine learning is a branch of artificial intelligence focus on developing algorithms that improve through experience automatically, and it is one way to address such ill-posed system identification problem. Since the mathematical algorithms are generic, it is inherently an interdisciplinary field. Many learning theories are built on concepts from probability, statistics, information theory, philosophy and have been successfully applied on a wide variety of engineering fields. In [4], the author speculated that the following four research areas might produce dramatic improvements in the state of the art of machine learning, (1) incorporation of prior knowledge with training data, (2) lifelong learning, (3) machine learning embedded in programming languages, (4) machine learning for natural language. As mentioned, it forms a core part of data mining techniques and is the focus of the work we present. In particular, we will focus on incorporating prior

knowledge and data, and creating a flexible and efficient learning application that can be embedding in existing systems and databases.

1.2 Objective

With these motivations in mind, we now state the goal: we will develop a computational framework to support data mining and modeling of large-scale systems. To achieve this goal, new algorithms are proposed, an efficient and flexible learning framework is implemented, and the algorithms and the application are exercised on both benchmark and practical problems. From the view of algorithm development, an efficient algorithm that is robust against imperfect data, solves an inverse modeling problem in reasonable time, and combines observations and prior knowledge within a discriminative learning approach, is the key of this framework. From the software design aspect, a loosely-coupled architecture is carefully designed. The implementation allows domain experts to contribute their specialties without modify or recompile the core program.

1.3 Challenges

How to make sense of the data? How data can be used to leverage modeling? How domain knowledge can be used to help interpret the data? How to incorporate domain knowledge? Before trying to answer these questions, it is useful to understand the difficulties behind the problem. First, as mentioned, unprocessed data is not useful per se. To support decision making, what we need is the information and knowledge hidden in the data. That is, to summarize the similarities, differences, and relationships discovered in the data within a specific context, and then turn this information into knowledge, i.e., effective procedures that achieve particular results. Second, the sheer volume of data collected from a network of sensors often poses a great challenge to computation resources. Even with a modern high-performance computer, an efficient algorithm that can be solved in polynomial time is necessary for most nontrivial problem. Third, even through the available data sets may be large, they often comprise only a sample from the complete population. To obtain a realistic understanding of the underlying system, we need to build a model that generalized from the sample to the population. Fourth, data available for analysis is often imperfect. Data collected through scientific equipments may suffer from various sources of noises, and result in incomplete and inaccurate measurements. Data collected by human, or data related to human behaviors, is obviously problematic: it is often biased by subjective, anecdotal experiences and is prone to error. Fifth, to better understand the system that generated the data and the interactions between its components, it is often necessary to build a model representing the underlying system. However, to build a model from the observed data is an ill-posed inverse problem. Sixth, in addition to the observed data, we often have some knowledge about the system of interest. How to incorporate the domain knowledge with the data, so the model can take advantage of both factors, is still an area of active current research. Last but not least, a generic computational framework built using modern programming architecture to support researches in different fields is still an unfulfilled need.

1.4 Proposed Approach and How It Meets the Challenges

1.4.1 Learning as a Model Building Process

The concept of modeling and simulation has been extensively applied in both the engineering and business world. For instance, when designing a high-rise building, a digital integrated circuit, or a new business strategy, a computer model and simulation is usually developed and tested beforehand. It is because the cost and risk of considering the system design trade-offs in the early design stage is significantly lower, and it also help us to better understand the system, optimize the performance and the reliability. Since a simulation is the execution of the model, and it is more straightforward once we have had a realistic model, we will focus on the development of models instead of simulations.

As mentioned, although can be very accurate, an analytical model such as a finite element model of a structure, it is often time-consuming and requires extensive domain knowledge to build. Moreover, in a dynamic system, states and properties of the system may change over time and a model needs to be updated frequently. This too requires significant amount of effort and makes an analytical model difficult to maintain. On the other hand, a probabilistic-based proximate model is an attractive alternative. We present a framework that generates a proximate model through machine learning. The work we are presenting is build on top of a recently developed discriminative learning algorithm, support vector machine. Discriminative learning algorithms such as SVM and boosting [5] have become standard techniques of applied machine learning, because they have achieved record benchmark results in a variety of domains. In addition to performing pattern recognition tasks, the goal, however, is to build a statistical model through the discriminative learning process.

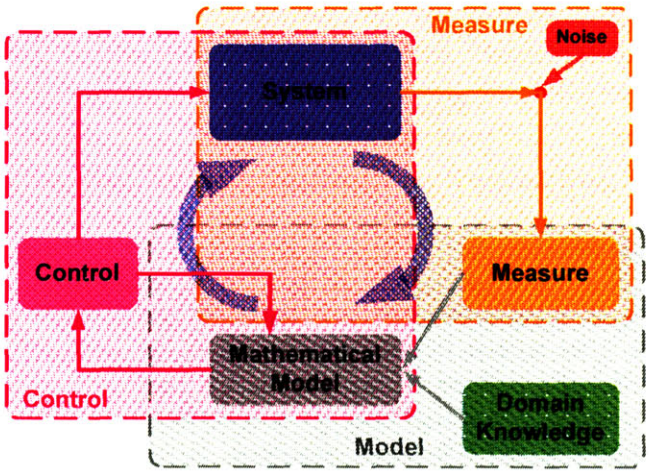


Figure 1.3 Creating Statistical Model using Measurements and Domain Knowledge

However, a statistical model requires training, and is prone to error when the available data are inaccurate or incomplete. As depicted in figure 1.3, we propose a framework that takes both data and domain knowledge into account and can generate a model that is efficient and robust.

1.4.2 Incorporating Prior Knowledge

Depending on how an algorithm models the underlying probability distribution of a system, the learning algorithm can be categorized into one of the two groups, discriminative or generative learning. Generally speaking, generative approaches make assumptions on the underlying probability distribution of the system and select the one which best describes the data; while discriminative approaches make no effort to discover the probability distribution, rather, they try to find a model that is best consistence with the data directly. Examples of generative learning include Bayesian learning networks and Markov models, and discriminative approaches include artificial neural networks, nearest neighbors, and support vector machines. Generally speaking, generative approaches need fewer training data and are less sensitive to data quality because of the utilization of prior knowledge, and discriminative approaches are leading in both performance and accuracy.

Discriminative learning algorithms assume the learning algorithm has no prior knowledge except for the representation of hypotheses, and that it must learn solely from training data. It is well understood that prior knowledge can reduce sample complexity and improve learning accuracy. Therefore, we incorporate prior knowledge in every stage of the learning process, as shown in figure 1.4, and bridge the two paradigms.

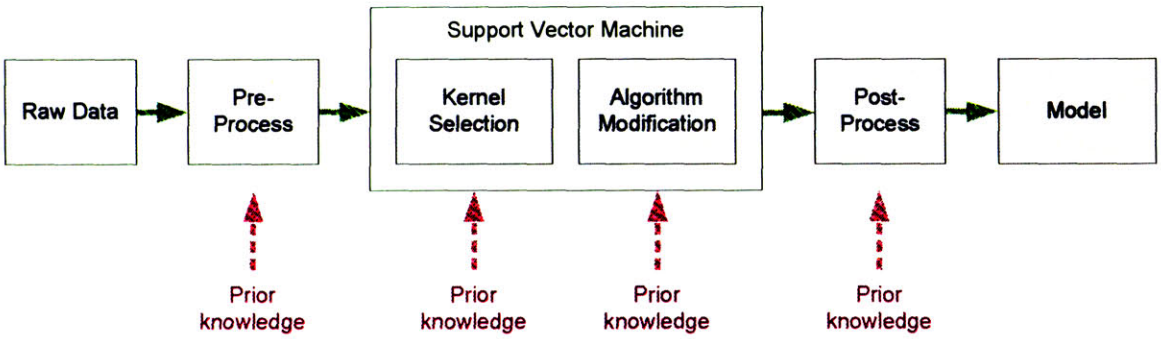


Figure 1.4 Incorporating Prior Knowledge into Discriminative Learning

1.4.3 Solving Large-Scale Problem

Many learning algorithms show promising results in testing problems failed miserably when dealing with large-scale problem. One reason is often referred to as the curse of dimensionality, which occurs as the dimensionality of a problem is increased, and it imposes serious limitation to many of the learning algorithms. Support vector machines get around the problem by using kernel functions that can be evaluated implicitly and efficiently. However, two problems remain unsolved by the kernel methods. First, not all kernels can be evaluated implicitly, thus a computational efficient method is required for those kernels. Second, when dealing with large-scale problems, the number of training samples also imposes a great challenge on computing resources. Mathematically, training SVM is equivalent to solving a quadratic programming problem with linear constraints, and the number of variables is equal to the number of training data points. Solving QP problem has been studied for a long time, and there are many general purpose algorithms available. However, the optimization problem becomes challenging when the number of variables exceeds several thousands, and old existing solvers can not handle the size of a practical machine learning problem. For instance, gradient-based solver typically requires n^2 memory space to store the matrix and $\Theta(n^3)$

to decompose the matrix, where n is the number of training data points. Therefore, if a solver can solve a problem with 1,000 samples in 10 seconds, the same solver will need 25 days to solve a problem with 60,000 samples.

The framework takes advantage of the fact that the number of support vectors is often much lower than the number of total training examples. For instance, when performing handwriting recognition on MNIST dataset with linear kernel results in a number of support vector to number of training samples ration around 0.2. The number is larger for noisier problem and is lower for simpler problems. By re-sampling the training dataset and solve the quadratic programming problem separately, and combining the support vectors, a much more efficient solver can be obtained. Also, since the process is highly independent, each subset can be solved in parallel and further increase the performance.

1.4.4 Simplifying Multidisciplinary Cooperation

The proposed framework is implemented as a Microsoft .NET assembly, and it is both language and platform independent. Also, it can be embedded in database, integrated into existing systems, and exposed as web service, as we will show in later chapters. As mentioned, a major objective of this work is to allow domain knowledge to be integrated into discriminative learning, and is fulfilled by kernel function and a loosely-coupled architecture. Support vector machine decouples kernel selection from the learning algorithm, allowing them to be modified separately by researcher with different background and interests. This is also a design goal of the architecture. Through remoting, user defined new kernels can be written separately and invoked by the application at run-time. This allows the domain experts to write customized kernels without rewriting and recompiling the core application. Also, any change made in the framework, for instance, updates on the optimization solver, will not affect the domain specific implementations. Hence, the loosely-coupled structure simplifies cooperation among different disciplines, and makes the framework easy to maintain and scalable.

1.5 Summary

The goal of this study is to provide a generic learning framework to boost the probabilistic modeling and analysis process, and apply it to large-scale, real-word problems. The goal is fulfilled by the contributions in three different levels, as shown in figure 1.5.

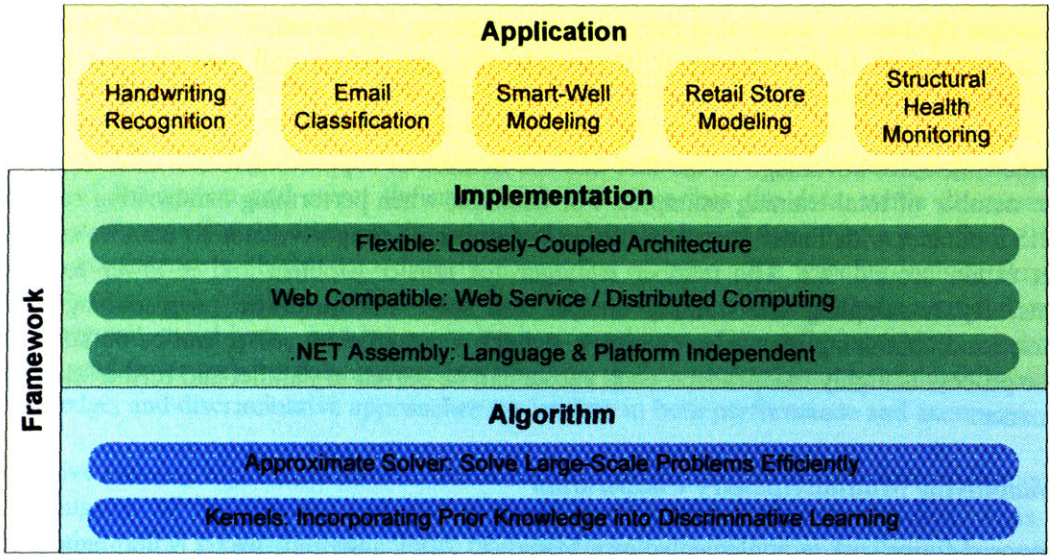


Figure 1.5 Contributions

The proposed algorithm takes both domain knowledge and data measured from the system into consideration and provide a mechanism that combines domain knowledge and statistical learning. An efficient approximation solver is suggested that allow the algorithm to solve large-scale problem efficiently. The framework is built on a serviced-oriented architecture that provides a loosely-coupled, robust, and highly interoperable software system. Also, several benchmark problems and a practical engineering are used to validate the methodology.

In particular, I will build black/gray box models through learning, reduce model size using kernel based feature selection, incorporate with other techniques such as wavelet to pre-process the data, and develop domain specific kernels that utilize the prior knowledge of the problems to guide the learning. I also implement my own library of support vector machine in modern network-aware objective-orientated programming language, and scale it up by XML web service and distributed computing. Last but not least, we will apply the approach to solve large scale problems in document classification, structural health monitoring, permanent down-hole gauge data interpreting.

The first contribution is to provide a general purposed kernel that can be used to introduce existing domain knowledge into support vector machine learning. The second contribution is to provide an efficient solver that can handle very large data sets. It takes advantages of the sparseness of support vectors and be parallelized easily to further speed-up the computation. The third contribution is to provide a robust, flexible, yet lightweight implementation of the proposed algorithm. It is written in a platform independent language and can be integrated into existing systems, or embedded into database easily. Further, the implementation is compatible with modern web protocols, and can be exposed as a web-service, or as an intelligent kernel of a larger service. The fourth contribution is to apply the proposed framework on an engineering problem.

The objective of this work is two-fold. On the business side, we want to provide a flexible virtual system that can conduct experiments that are expensive or impossible to perform field tests, help evaluating the value of new process changes in the system, and optimize the system through a measure-model-control loop. On the technological side, we want to deliver a hybrid model utilizes

the best of algorithms drawn from system dynamics and machine learning and develop a robust system to incorporate learning with modeling and simulation. In this work, we present such a system, along with the algorithms, design issues, and experiments of large scale, real world problems.

1.6 Roadmap

We have given a high-level introduction in the present chapter to the problems we are trying to address and the approaches we propose to meet the challenges. In chapter two, we will review and introduce the fundamental theory in machine learning. It provides the context within which the work takes place. Chapter three introduces the recent developed machine learning technique, support vector machine. Successful applications have shown that support vector machines not only have a more solid statistical foundation than artificial neural networks, but it also outperforms artificial neural networks in a wide variety of fields, in terms of both speed and accuracy. For this reason, support vector machine is used as the foundation of the work. Support vector machine is often explained through one of the two different views, 1.) regularization and functional analysis, 2.) geometry and optimization. We opt for the later for pedagogical reasons but the first view is also mentioned when necessary. Chapter four discusses the general properties of kernels, and the proposed approach of incorporating prior knowledge into support vector machine. Chapter five describes an approximate approach of solving the optimization problem in support vector machine. In chapter six, we apply the proposed approach on several practical problems and show that the approach is both effective and efficient. A summary is given in chapter 7.

Chapter 2 Machine Learning

Machine learning is the study of computer algorithms that improve their performance through experience. A broad definition adopted from [6] is listed below,

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if the performance at tasks in T, as measured by P, improves with experience E.

Therefore, a learning problem consists of a source of experience, usually in the form of observed examples; a class of tasks; and a measure of performance to be improved. Taking a handwriting recognition problem for example, the experience is the set of handwritten words with known categories, the task is to recognize and classify handwritten words from images into correct categories, and the performance measure is the percent of words classified correctly. Although machine learning is usually considered a subspecialty of artificial intelligence, it also draws concepts from information theory, neurobiology, philosophy, especially the study of probability and statistics. The major difference is, machine leaning not only consider the patterns of the data, it also concerns with the algorithmic complexity of computational implementations.

In this chapter, we review several important machine learning subjects that are related to the main topic of this research. Section 2.1 introduces the notations, general concepts, and measurements of machine learning. Section 2.2 explains Bayesian learning, the most important concept behind generative learning, which also provides the background knowledge when we introduce prior knowledge into SVM later. Section 2.3 explains the statistical learning theory, which gives the statistical foundation of SVM, which we will discuss in depth in chapter 3.

2.1 Learning from Examples

The goals of most traditional programming applications are to find outputs based on inputs. For instance, a structure analysis program can give us the response of the building under a specific load, given the dimensions, materials, and loadings of a building. Or the very basic Newton's second law: $F = m \cdot a$, which gives the acceleration of an object when the mass and external force are given. In this kind of traditional programming, we know the relations between inputs and outputs and those relations can be expressed in mathematics forms explicitly, as illustrated in figure 2.1.

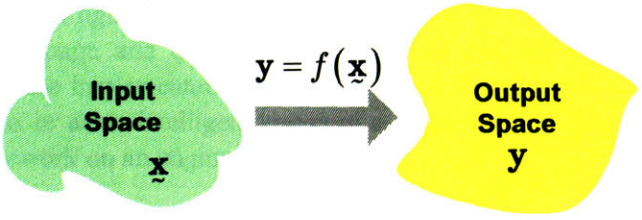


Figure 2.1 Traditional Programming.
X and f(X) are known. Given an X we can calculate Y from f(X)

In many practical problems the underlying functions that map from inputs to outputs are unknown. Frequently the inputs are noisy or part of the inputs is missing, and there is no guarantee that there is an underlying function that correctly maps the inputs to outputs. Therefore, it is natural to try to develop an algorithm that can learn how to model and recognize patterns from observed examples, much like the way humans learn new things. As depicted in figure 2.2, the goal of learning is to find a mapping function whose outputs "best described" the known outputs. Of course, we will give more rigorous definitions of what we mean by "best describe" when we introduce learning algorithms.

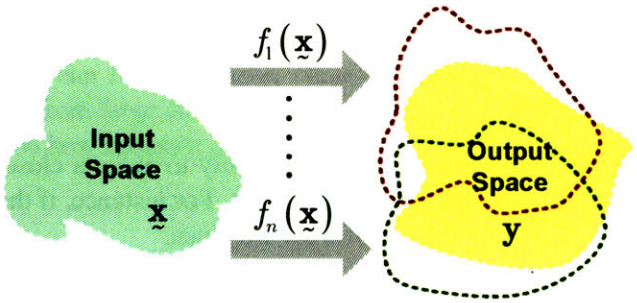


Figure 2.2 Machine Learning.
X and Y are known, but the mapping between X and Y is unknown

Since the task now is to determine causes for an observed effect or calibrate the parameters of a mathematical model to reproduce the observations, it is known as inverse problems. Inverse problems are usually not fulfill Hadamard's postulates of well-posedness [7], and thus are ill-posed. A well-posted problem must have the following three properties,

1. A solution exists
2. The solution is unique
3. The solution depends continuously on the data, in some reasonable topology.

Machine learning problems might not have a solution in the strict sense, and their solutions might not be unique and might not depend continuously on the data. For a long time, mathematicians focused mainly on well-posed problems because they felt that ill-posed problems cannot describe real phenomena and objects. Further, ill-posed problems, usually needs to be solved numerically, need additional information about the solution, such as an assumption on the smoothness or a bound on the norm, (known as regularization, for instance, the widely used Tikhonov regularization), in order to get a stable solution. However, ill-posed inverse problems have important applications in many engineering fields, and are the only solutions to the tasks.

2.1.1 Machine Learning as a Search Problem

As illustrated in Figure 2.2, given a training data set $\mathcal{D} = \{(\underset{\sim}{x}_i, y_i), \cdots, (\underset{\sim}{x}_m, y_m)\} \in \underset{\sim}{x} \times y$, and denote the function space, or hypothesis spaces, as $\mathcal{H}$; each candidate function, or hypothesis, in the function space by h, a machine learning task can be thought of as to find a $h \in \mathcal{H}$ that best describes the underlying sample generation function. In other words, a machine learning task can be thought of as a search problem in the hypothesis space. As a search problem, the basic concept of machine learning can be separated into two steps. First, we need to choose a set a candidate

functions (hypotheses function set, or hypothesis space, $\mathcal{H}$) to direct the learning process and limit the search space. Second, we select the best hypothesis from the hypotheses set base on how good it fits the training data. Clearly we need to make a few trade-off in each step. For instance, we would like to have a large hypothesis space to increase the probability of finding a best match, but we also want the algorithm to be computational efficient. We want to find a function that closely fit the training data set, but more importantly we want the function to be generalized well, i.e., it not only gives close matches to the training data but also gives accurate prediction on testing data. What we mean testing data is data drawn from the same distribution as the training data but not used in the training process, thus "unseen" by the algorithm.

2.1.2 Overfitting and Underfitting

The real power of learning lays on generalization, the ability to make a close estimation on unseen sample based on the rule learned from the given training set. For instance, if the task given is to learn the concept of "tree", and the following images of trees are given,

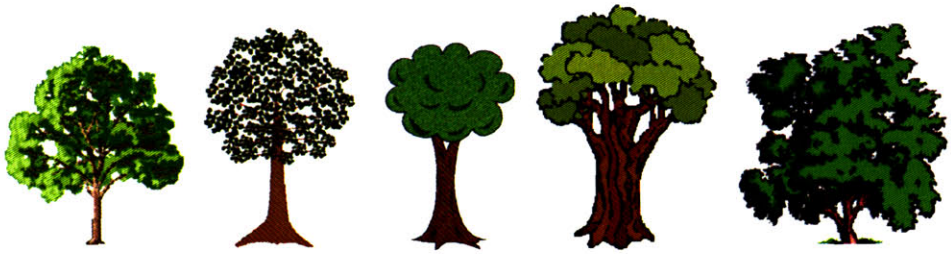


Figure 2.3 Samples of Trees

Obviously we want our trained classifier to be able to know that an image represents a tree if that image is one of examples given to the classifier in training stage. Ideally, we want the classifier to tell that an new image also represents a tree if the given image is "similar" to the training images, for example, figure 2.4a; and the classifier should be able to tell what is not a tree, e.g. a green truck shown in figure 2-4b. An overfitted classifier thinks figure 2.4a is not a tree because it has fewer leaves than the one in figure 2.3. An underfitted classifier may think figure 2.4b is also a tree because it is green.

Figure 2.4a A tree.

2.4b A Non-tree Object

This issue has long been the considered by researchers in related fields. It is sometimes referred to as the *bias-variance dilemma* in statistics, or the choice between *estimation error* and *approximation error*. Take regression for example, when the training set (dots) in figure 2.5 is given, we can impose a restriction on the model and only allow linear functions to be used. If this choice is based on our

domain knowledge of the underlying problem, this restriction often helps the generalization ability. Otherwise it is a forced bias, and a restricted function space results in a higher approximation error because even the best possible solution cannot approximate the true dependency of the distribution. On the other hand, if a large class of functions is selected, e.g., a polynomial function with high degree or Gaussian radial basis functions with very small σ,

$$f(x) = \sum_{i=1}^{n} a_i \exp\left(-\|x - x_i\|^2 / 2\sigma^2\right) \tag{2.1}$$

the large hypothesis space can fit the training samples perfectly and results in zero approximation error, but it often suffers from large statistical estimation error, i.e., large variance, and is subject to fluctuations, depending on how accurate the training samples are.

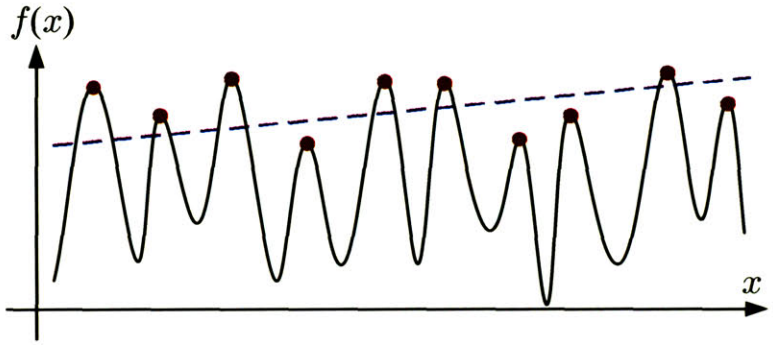


Figure 2.5 Overfitting and Underfitting

Now we have a basic idea of the problem, and we can give a formal discussion of the bias-variance trade off. Assume we have a training data set of m samples, $\mathcal{D}_k = \{(\underset{\sim}{x}_i, y_i), \cdots, (\underset{\sim}{x}_m, y_m)\}$, which is generated by a unknown underlying process $f_i = f(\underset{\sim}{x}_i)$. Because f_i is unknown, given an $\underset{\sim}{x}_i$ we can only measure the corresponding y_i, with an error ε_i, i.e.,

$$y_i = f(\underset{\sim}{x}_i) + \varepsilon_i \tag{2.2}$$

The distribution of ε_i is usually modeled as a Gaussian distribution with zero mean,

$$E(\varepsilon_i) = 0 \text{ and } Var(\varepsilon_i) = E\left((\varepsilon_i - 0)^2\right) = \sigma_\varepsilon^2 \tag{2.3}$$

this is a justifiable assumption when no other information is available. Since in many real situations it is possible to consider the error of an observation as the result of many (say, n) independent small errors, the distribution of ε_i will approach a Gaussian distribution when n approach infinity, according to central limit theorem.

For a specific set of training data, an approximation function $h_i = h_i(\underset{\sim}{x}_i)$ can be learned. To measure how good the approximation is, we can calculate the mean square error, MSE,

$$MSE(\mathcal{D}_k) = \frac{1}{m}\sum_{i=1}^{m}\left(y_i - h(\underset{\sim}{x}_i)\right)^2 \tag{2.4}$$

The reason MSE is so wildly used as an error measurement, historically, is because it is a convex function and has better mathematically property than the more intuitive absolute error, $\frac{1}{m}\sum_{i=1}^{m}\left|y_i - h(\underset{\sim}{x}_i)\right|$. However, MSE has a deep connection with maximum likelihood, which gives more statistical reason of choosing MSE as an error measurement, and we will discuss in more detail when we introduce Bayesian learning.

To assess the effectiveness of the approximation function *h*, we need to repeat the MSE calculation arbitrarily many times using test data drawn from the same unknown function *f*, that is, we want to take the expectation of the MSE with respect to training sets

$$E\left[MSE(\mathcal{D}_k)\right] = E\left[\frac{1}{m}\sum_{i=1}^{m}\left(y_i - h(\underset{\sim}{x}_i)\right)^2\right] = \frac{1}{m}\sum_{i=1}^{m}E\left[\left(y_i - h(\underset{\sim}{x}_i)\right)^2\right] \tag{2.5}$$

Note that h is learned from one specific training set $\mathcal{D}_k$, h is therefore a function of $\mathcal{D}_k$. Let us investigate the expectation inside the sum in equation (2.5),

$$\begin{aligned} E\left[\left(y_i - h_i\right)^2\right] &= E\left[\left(y_i - f_i + f_i - h_i\right)^2\right] = E\left[\left(\varepsilon_i + f_i - h_i\right)^2\right] \\ &= E\left[\varepsilon_i^2 + \left(f_i - h_i\right)^2 + 2\varepsilon_i\left(f_i - h_i\right)\right] \\ &= E\left[\varepsilon_i^2\right] + E\left[\left(f_i - h_i\right)^2\right] + 2E\left[\varepsilon_i\left(f_i - h_i\right)\right] \\ &= \text{Irreducible Error} + E\left[\left(f_i - h_i\right)^2\right] + 0 \end{aligned} \tag{2.6}$$

The term $E\left[\varepsilon_i\left(f_i - h_i\right)\right]$ in equation (2.6) equals to zero because

$$E\left[\varepsilon_i\left(f_i - h_i\right)\right] = E\left[\varepsilon_i f_i\right] - E\left[\varepsilon_i h_i\right] = f_i E\left[\varepsilon_i\right] - E\left[\varepsilon_i\right]E\left[h_i\right] = 0 \tag{2.7}$$

Note that in general $E[XY] \neq E[X]E[Y]$ unless X and Y are probabilistically independent. The noise ε_i is a property of the way we measure the data, which is independent of the approximation function h. The term $E\left[\left(y_i - h_i\right)^2\right]$ can be further decomposed,

$$\begin{aligned} E\left[\left(f_i - h_i\right)^2\right] &= E\left[\left(f_i - E[h_i] + E[h_i] - h_i\right)^2\right] \\ &= E\left[\left(f_i - E[h_i]\right)^2\right] + E\left[\left(E[h_i] - h_i\right)^2\right] + 2E\left[\left(f_i - E[h_i]\right)\left(E[h_i] - h_i\right)\right] \\ &= \left(f_i - E[h_i]\right)^2 + E\left[\left(E[h_i] - h_i\right)^2\right] + 0 \\ &= \text{Bias}^2(h) + \text{Variance}(h) \end{aligned} \tag{2.8}$$

Note that,

$$\begin{aligned} E\left[\left(f_i - E[h_i]\right)\left(E[h_i] - h_i\right)\right] &= E\left[f_i E[h_i]\right] - E[f_i h_i] - E\left[E[h_i]^2\right] + E\left[E[h_i] h_i\right] \\ &= f_i E[h_i] - f_i E[h_i] - E[h_i]^2 + E[h_i]^2 \\ &= 0 \end{aligned} \tag{2.9}$$

Thus, the effectiveness of the approximation function h is consisted of three parts,

$$\frac{1}{m}\sum_{i=1}^{m} E\left[\left(y_i - h(\underset{\sim}{x}_i)\right)^2\right] = \text{Irreducible Error} + \text{Bias}^2(h) + \text{Variance}(h) \tag{2.10}$$

The sources of errors is summarized in table 2.1, where irreducible error comes from noise, approximation error (bias) is incurred by imposing restriction on hypothesis space $\mathcal{H}$, and estimation error (variance) indicates how well the learned hypothesis h performs comparing to the best possible $h \in \mathcal{H}$.

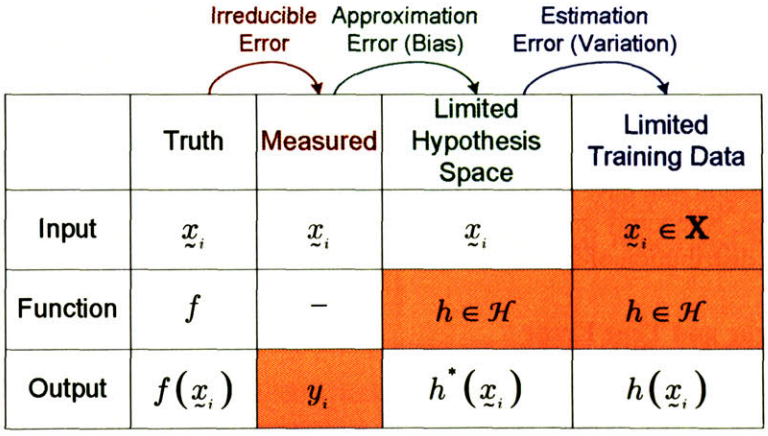


Table 2.1 Decompose the Source of Errors

We want to find an h that minimizes the overall error, i.e., the sum of approximation error and estimation error. To decrease approximation error we need to increase the size of hypothesis space $\mathcal{H}$. However, given a limited number of training samples, finding the best h in a larger $\mathcal{H}$ is more difficult, and results in a higher estimation error, i.e., overfitting. Overfitting not only happens in simple regression problems, it is also a major issue in much more complicated machine learning algorithms, such as neural networks. A great deal of research has been conducted to address this problem, mostly heuristics, to help design a neural network not to overfit a given data set. Frequently used techniques include selecting less hidden units, limiting the hypothesis function space, and stopping the training procedure earlier to avoid overfitting.

2.2 Statistical Learning Theory

As discussed in section 2.1.1, given a training data set, a function space to search, and a performance measure, a machine learning task can be thought of as a search problem in the function space. The question becomes how to choose a performance measure? How to choose a function (hypothesis) space? And how much training data do we need? To answer these questions, an overview of some of the fundamental learning theories is given in this section.

2.2.1 Sampling

Denote the target function that our learning algorithm is called upon to learn as f, and the set of all possible instances over which the target functions may be defined as P. In other words, P is the whole population of samples of f, and since P contains all possible instances generated using f, learning P is equivalent to leaning f. When we are trying to learning something about some aspects of P, we are actually learning from samples collected from P, $\underset{\sim}{x}_1,...,\underset{\sim}{x}_m \in \mathbf{X}$, because the size of P is usually infinity. We often assume the samples are drawn from P independent and identically, i.e., every time when we draw a sample from P, we draw the sample randomly following the same unknown but stationary probability distribution $\mathcal{D}$. This is referred to as drawing an IID (independent and identical distributed) sample from P. For example, suppose we want to learn the relation between "people who have advanced degrees" and their height, weight, and age. In this case, P represents the set of all people, and each sample $\underset{\sim}{x}$ has the format of (*height*, *weight*, *age*). The goal is to find a function f that can predict whether a person has an advanced degree or not based on his or her height, weight, and age. To obtain enough samples, we need to interview some people from the whole population. Randomly interviewing people who walk out of the T station at Kendall square can be one specific distribution of instances $\mathcal{D}$, although it is obviously biased.

2.2.2 Training Error and True Error

The *training error* of a classification task can be easily defined by the number of training samples that are misclassified by the learned hypothesis. If a classifier has a zero training error, it is often referred to as a consistent classifier. It is worth mentioning that a consistent classifier is not necessary better than a non-consistent classifier, because what we expect from a good classifier is one that also performs well in the unseen data, not just the training samples. In other words, we want to find a classifier with a small true error.

After a hypothesis is learned, we want to know how it performs against future instances drawn from the population according to the same probability distribution. The real expect error rate, or the *true error*, can be defined as following.

> The true error of hypothesis h with respect to target function f and distribution $\mathcal{D}$ is the probability that h will misclassify an instance drawn at random according to $\mathcal{D}$.

We can calculate the true error the same way we calculated the training error, provided we have samples from every point in the entire population; not just the training samples drawn from the population. The comparison of training error and true error is show in figure 2.6.

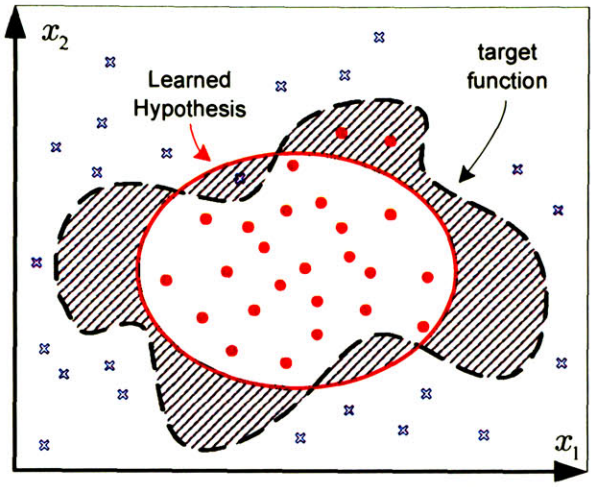


Figure 2.6 Training Error and True Error

There are 2 positive samples (shown as disks) classified as negative, and 1 negative (shown as cross) classified as positive by the learned hypothesis, h. Therefore, the training error in this case is 3/48 = 6.25%, where 48 is the total number of samples. Similarly, the true error is the area where the learned hypothesis disagrees with the target function (the shaded area) divided by the total area, assuming the distribution $\mathcal{D}$ is a uniform distribution. Note that the true error is highly dependent on the underlying distribution $\mathcal{D}$. Suppose $\mathcal{D}$ is not a uniform distribution and it has higher probability density in the misclassified region, the true error will be higher. The training error also depends on $\mathcal{D}$, because the samples are drawn IID according to $\mathcal{D}$. If $\mathcal{D}$ has higher probability density in the misclassified region, then more training samples will be from that region and results in a higher training error for the specific hypothesis.

We are interested in the true error while we can only get the training error. The question is, is there a relation between the two? Or better yet, can we bound the true error by using the training error? The answer is yes. In the next section we will answer the following question, how many training samples are sufficient for a hypothesis to guarantee to have a small true error? We will introduce a few terminologies and restate our question more formally later.

2.2.3 Sample Size and Error Bound

How many training samples do we need to achieve a desired accuracy, i.e., guarantee a good generalization? Before answering this question, we first define a measure of the goodness. With finite number of training samples, it is not realistic to ask a learning algorithm to produce a hypothesis that has zero error for all future instances. To relax our expectation a little, we want the learned hypothesis to be able to generate a close approximation (with testing error less than ε), most of the time (with probability greater than $1-\delta$). This is often referred to as the PAC (probably approximately correct) learning. Both ε and δ can be arbitrarily small; if we set both to be zero then it is equivalent to the "always correct" case. The question can now be restated as, how many training samples do we need to achieve a PAC result? Since we are searching within $\mathcal{H}$, it is obvious that the number will depend on the complexity of $\mathcal{H}$. A proof assuming $\mathcal{H}$ contains at least one consistent learner is give below.

A learning task can be viewed as a search problem in the hypothesis space. In a classification problem, assume we have a set of finite numbers of IID training data, and the goal is to learn the underlying discrimination function, h. Because the discrimination function is unknown, without other prior knowledge, we can only select a general hypothesis space $\mathcal{H}$ and hope that h is in $\mathcal{H}$, or can be closely described by a function $h \in \mathcal{H}$. If an h can classify all training samples correctly, it is called a *consistent classifier*, or consistent learner. For a given $\mathcal{H}$, it may contain none or many consistent learners. Note that all consistent learners are equally well because they all fit the training samples correctly, i.e., have zero training error. The set of all consistent learners form a subset of $\mathcal{H}$, known as version space, *VS*, of hypothesis $\mathcal{H}$ and training samples $\mathcal{D}$. As mentioned before, a consistent classifier has zero training error but there is no guarantee that its true error is also small. In the following, we will prove that under some general assumptions, when certain numbers of training samples are available, every consistent classifier is guaranteed to have a small true error.

Now we can restate the goal one last time as, how many training samples are required such that all consistent learners in the hypothesis space are PAC? Assume we have a hypothesis space $\mathcal{H}$, which contains $|\mathcal{H}|$ different hypotheses. Some of the hypotheses are consistent, some are not. Assume the worst k of the $|\mathcal{H}|$ hypotheses have true error greater than ε when m training samples are used. Then, to prove that all consistent learners have true error less than ε is equivalent to prove that none of the k worst hypotheses is consistent.

Since we know the true error is at least ε for the worst k of the $|\mathcal{H}|$ hypotheses, the probability of randomly pick an instance and it is classified correctly by one of the classifier is at most $1-\varepsilon$. The probability of all m training samples are classified correctly (a consistent learner) is then at most $(1-\varepsilon)^m$. That is the probability for any one of the k hypotheses to be consistent. Hence the probability of none of the k hypothesis is consistent is $\left[1-(1-\varepsilon)^m\right]^k$, and the probability that the version space is not ε-exhausted, i.e., there exist at least one of the k learner is consistent, is $1-\left[1-(1-\varepsilon)^m\right]^k$.

When m is large, the probability of $(1-\varepsilon)^m = p \cong 0$. For any small number p we know,

$$\begin{aligned} f(p) &= 1-(1-p)^k = f(0)+f'(0)p+\frac{1}{2!}f''(0)p^2+\cdots \\ &= 0+kp-k(k-1)p^2+\cdots \\ &\le kp \end{aligned} \tag{2.11}$$

Therefore,

$$1-\left[1-(1-\varepsilon)^m\right]^k \le k(1-\varepsilon)^m \tag{2.12}$$

Also, since $k<|\mathcal{H}|$ and $1-\varepsilon \le e^{-\varepsilon}$, we have,

$$k(1-\varepsilon)^m \le |\mathcal{H}|(1-\varepsilon)^m \le |\mathcal{H}|e^{-\varepsilon m} \tag{2.13}$$

We want this probability to be small, say, less than δ,

$$|\mathcal{H}|e^{-\varepsilon m} \le \delta \tag{2.14}$$

Rearrange (2.14) we have,

$$m \ge \frac{1}{\varepsilon}\left(\ln|\mathcal{H}| + \ln(1/\delta)\right) \tag{2.15}$$

Equation (2.15) is based on the assumption that at least one hypothesis in $\mathcal{H}$ consistent, i.e., has zero training error. A more general proof that does not has this constraint can be obtained by using the general Hoeffding bound,

$$P\left(error_{true}(h) > error_{training}(h) + \varepsilon\right) \le e^{-2m\varepsilon^2} \tag{2.16}$$

For all $h \in \mathcal{H}$, we have

$$P\left(\exists h \in \mathcal{H},\ error_{true}(h) > error_{training}(h) + \varepsilon\right) \le |\mathcal{H}|e^{-2m\varepsilon^2} = \delta \tag{2.17}$$

Rearrange (2.17) we have,

$$m \ge \frac{1}{2\varepsilon^2}\left(\ln|\mathcal{H}| + \ln(1/\delta)\right) \tag{2.18}$$

Or equivalently,

$$error_{true}(h) \le error_{training}(h) + \sqrt{\frac{\ln|\mathcal{H}| + \ln(1/\delta)}{2m}} \tag{2.19}$$

Note that (2.15) and (2.18) have a similar format that is also very intuitive. More training samples are required if the hypothesis space is larger, or a tighter PAC result is required (smaller true error and/or higher probability). Although intuitive, the major parameter $|\mathcal{H}|$ is not a good measurement because it is usually not countable. A better metric for the complexity of the hypothesis space $\mathcal{H}$ is Vapnic-Chervonenkis dimension (VC dimension, or $|\mathcal{H}|_{VC}$). The proof using VC dimension can be found in [8]. Here we state without proof,

$$error_{true}(h) \le error_{training}(h) + \sqrt{\frac{|\mathcal{H}|_{VC}\left(\ln\left(2m/|\mathcal{H}|_{VC}\right)+1\right) + \ln(4/\delta)}{m}} \tag{2.20}$$

The structure of (2.20) is similar to (2.19) and the above discussion also applies. Now we give a definition to VC dimension. A hypothesis space $\mathcal{H}$, or function family, is said to shatter a set of data

points $\mathcal{D}$ if for all assignments of labels to the points in $\mathcal{D}$ there exists a hypothesis h such that h correctly classifies all points in $\mathcal{D}$. VC dimension of a hypothesis space $\mathcal{H}$ is defined as the maximum number of points that can be shattered by $\mathcal{H}$. Note that m points can be arranged in many different ways, and the VC dimension equals to m if any of the arrangement can be shattered by $\mathcal{H}$. A simple illustration of VC dimension is shown in figure 2.7 ~ 2.9,

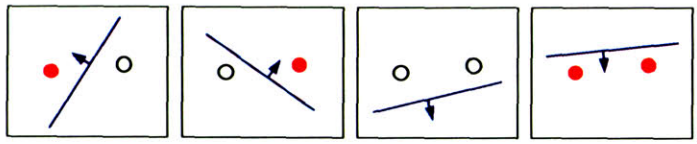


Figure 2.7 Two points can be shattered by a line in 2D

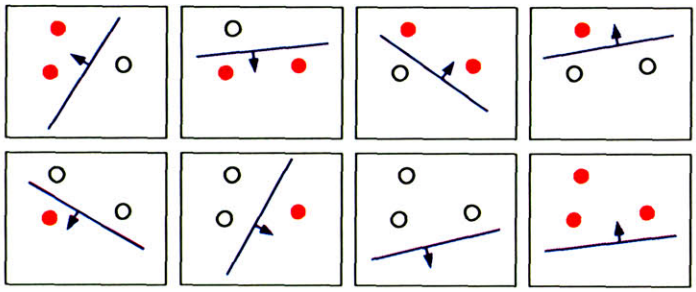


Figure 2.8 Three points can be shattered by a line in 2D

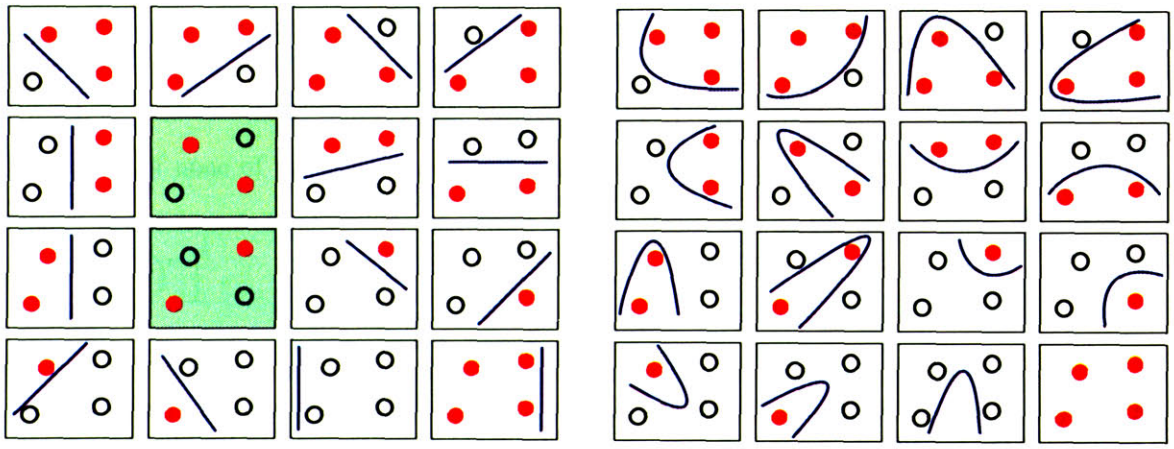


Figure 2.9 A more complex function family is required to shatter 4 points in 2D

VC dimension plays an important role in the theory of support vector machine and we will discuss in more detail later.

2.2.4 Performance Measure

Mean square error is the most widely used error measure. Minimizing mean square root is widely used in every engineering field, yet it is usually taken as granted. Before we can discuss the performance measure, we need to introduce Bayes' rule. Some necessary notations are required before we give a precise definition to Bayes theorem. Denote P (A = *true*), or P (A), as the probability that event A is true in a world where event A exists, as shown in figure 2.10. The

probability that event A is false is then $P(\neg A) = P(\text{World}) - P(A) = 1 - P(A)$. Similarly, $P(B)$ is the probability that event B is true, and $P(A \wedge B)$ is the probability that event A and event B are both true, and therefore called *joint probability*. $P(A)$ and $P(B)$ is known as *marginal probability*, meaning that the probability of the event is regardless of other events.

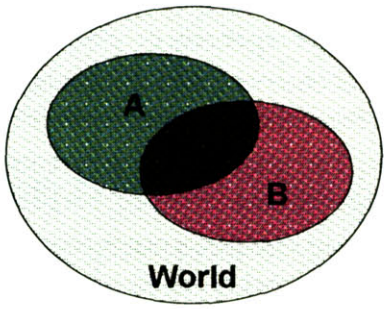


Figure 2.10. Marginal Probability, Joint Probability, and Conditional Probability

Denote $P(A = true \mid B = true)$, or $P(A \mid B)$, as the probability that event A is true given that event B is also true. This is referred to as *conditional probability*, and read as the probability of A given B. Given this definition, we can now express conditional probability using joint the joint probability and marginal probability, as shown in (2.21).

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)} \tag{2.21}$$

Equation (2.21) can be read as "The probability of event A is true given that event B is true equals to the probability of both event A and B are true in a world where event B is true". In machine learning context, we are often interested in determining the *best* hypothesis h from a hypothesis space $\mathcal{H}$, given some observed data $\mathcal{D}$ from the entire population $\mathcal{D}$. Equation (2.21) given a natural definition of what we mean by best, given $\mathcal{D}$ we want to find the h that maximize $P(h \mid \mathcal{D})$. It is not very useful if we use (2.21) directly because we do not know $P(h \wedge \mathcal{D})$. However, since

$$P(B \mid A) = \frac{P(A \wedge B)}{P(A)} \tag{2.22}$$

and it shares the same term $P(B \mid A)$ with (2.21), we can obtain the following by combining (2.21) and (2.22),

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \tag{2.23}$$

Equation (2.23) is known as *Bayes theorem*. If we put Bayes theorem into machine learning context, and substitute A by h and B by $\mathcal{D}$, we can rewrite (2.23) as

$$P(h \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid h) P(h)}{P(\mathcal{D})} \tag{2.24}$$

which states that the probability of hypothesis h given observation data $\mathcal{D}$ equals to the probability of observing data $\mathcal{D}$ given a world where the hypothesis h holds true, multiply by the probability of h is true, and divided by the probability of observing $\mathcal{D}$. $P(\mathcal{D} \mid h)$, called the *likelihood* of the data $\mathcal{D}$ given h, represents the probability of observing data $\mathcal{D}$ if the hypothesis h is in fact the underlying data generating process. $P(h)$ is often called the *prior* probability of h, and it reflects our prior belief of the chance that h happens. The reason it is a prior knowledge is because it represents a value we know before we see any observed data. $P(h \mid \mathcal{D})$, on the other hand, is called the *posterior* probability of h, because it represents our confidence that h happens after we have seen data $\mathcal{D}$.

Recall that in machine learning, the goal is, given a set of observed data, to find a hypothesis that is most likely to be the true underlying data generating process. This is equivalent to find a $h \in \mathcal{H}$ that maximizes the probability $P(h \mid \mathcal{D})$. Denote such a hypothesis as h_{MAP}, a maximum a posterior (MAP) hypothesis,

$$h_{MAP} = \arg\max_{h \in \mathcal{H}} P(h \mid \mathcal{D}) \tag{2.25}$$

Alternatively, we can find a hypothesis that maximizes the likelihood of data $\mathcal{D}$. It is called the maximum likelihood (ML) hypothesis, h_{ML},

$$h_{ML} = \arg\max_{h \in \mathcal{H}} P(\mathcal{D} \mid h) \tag{2.26}$$

The relation of h_{MAP} and h_{ML} can be established by using Bayes' rule,

$$h_{MAP} = \arg\max_{h \in \mathcal{H}} P(h \mid \mathcal{D}) = \arg\max_{h \in \mathcal{H}} \frac{P(\mathcal{D} \mid h) P(h)}{P(\mathcal{D})} = \arg\max_{h \in \mathcal{H}} P(\mathcal{D} \mid h) P(h) \tag{2.27}$$

Note that in the last step in (2.25) we dropped the term $P(\mathcal{D})$ because it is independent of h. If the prior probability $P(h)$ is not available, we often assign a constant to all hypotheses $h \in \mathcal{H}$, and the only term left in (2.25) is $P(\mathcal{D} \mid h)$. In other words, without any prior knowledge, MAP reduces to ML and $h_{MAP} = h_{ML}$. It is important to note that the goal of machine learning is (2.25) rather than (2.26), although sometimes we have to settle for (2.26), when no prior knowledge is available.

Now we are in the place of discussing performance measure. When a training data set, $\mathcal{D} = \{(\underset{\sim}{x}_1, y_1), \cdots, (\underset{\sim}{x}_m, y_m)\}$ is given, note that y_i is only a approximation of the true value $f(\underset{\sim}{x}_i)$. We assume y_i is normally distributed with average equal to $f(\underset{\sim}{x}_i)$ and standard deviation equal to σ_i, as shown in figure 2.11.

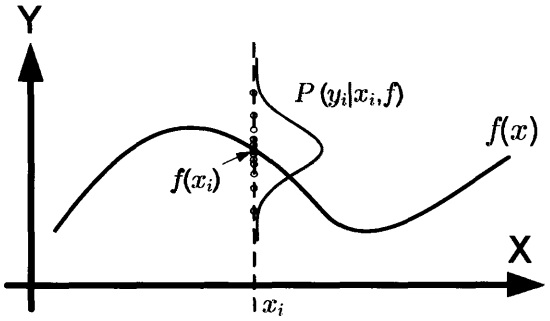


Figure 2.11 Error in Training Data

Loosely speaking, central limit theorem states that data which are influenced by many small and unrelated random effects are approximately normally distributed. Therefore it is justifiable to model errors as normally distributed when no other information is available.

Recall that the goal of machine learning is to find an $h \in \mathcal{H}$ that closest to the real, unknown, underlying data generating function f. Given a candidate h, the probability of observing one particular training sample is,

$$P(y_i \mid x_i, h) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(\frac{-(y_i - h(x_i))^2}{2\sigma_i^2}\right) \tag{2.28}$$

The probability of observing all training samples is,

$$P(\mathcal{D} \mid h) = \prod_{i=1}^{m} P(y_i \mid x_i, h) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(\frac{-(y_i - h(x_i))^2}{2\sigma_i^2}\right) \tag{2.29}$$

Take natural log (monotonic increasing) on both sides,

$$\log\left(P(\mathcal{D} \mid h)\right) = \sum_{i=1}^{m} \left(-\log\left(\sqrt{2\pi}\sigma_i\right) - \frac{(y_i - h(x_i))^2}{2\sigma_i^2}\right) \tag{2.30}$$

Maximize the log likelihood gives us minimize square root,

$$\begin{aligned} \arg\max_{h \in \mathcal{H}} \log\left(P(\mathcal{D} \mid h)\right) &= \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{m} \left(\log\left(\sqrt{2\pi}\sigma_i\right) + \frac{(y_i - h(x_i))^2}{2\sigma_i^2}\right) \\ &= \arg\min_{h \in \mathcal{H}} \sum_{i=1}^{m} (y_i - h(x_i))^2 \end{aligned} \tag{2.31}$$

We can see that minimizing square root is a special case of maximizing the log likelihood, i.e., it is the best one can have when there is no prior knowledge available.

2.2.1 The Structural Risk Minimization Principle and VC Dimension

We have seen that empirical risk minimization (ERM) can not guarantee a bounded true error because it does not take the model complexity into account. In contrast to ERM, which minimizes the empirical risk at any cost, Structural risk minimization (SRM) [9] describes a general principle looks for the optimal in trading off empirical error with hypothesis space complexity. SRM uses a set of models ordered in terms of their complexities. The complexity is generally given by the number of free parameters. VC dimension is a more accurate measure of model complexity. Model selected by using SRM corresponds to finding the model simplest in terms of hypothesis space and best in terms of empirical error on the data.

Considering a set S of nested sets of functions as shown in Figure 2.12, where $S_1 \subset S_2 \subset \cdots \subset S_n \subset \cdots$, each subset in S has a finite VC dimension $|\mathcal{H}|_{VC}$, and therefore $|\mathcal{H}_1|_{VC} \leq |\mathcal{H}_2|_{VC} \leq \cdots \leq |\mathcal{H}_n|_{VC} \leq \cdots$. An example of this kind of structure is the polynomial family of increasing order, or the set of splines with increasing nodes.

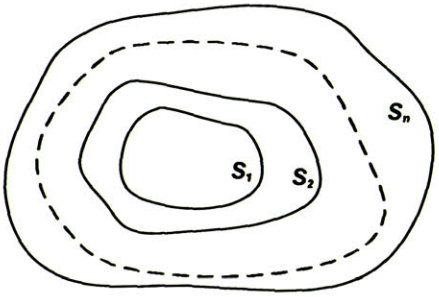


Figure 2.12 Nested Subset of Function (hypothesis) Spaces

Recall in Equation (2.20) we see that the real risk is bounded by the sum of empirical risk and the confidence term, which is a function of the VC dimension $|\mathcal{H}|_{VC}$. For each function space, we calculate the training error (empirical risk) and the confidence interval, and select the one that has the tightest bound on the actual risk, as shown in figure 2.13.

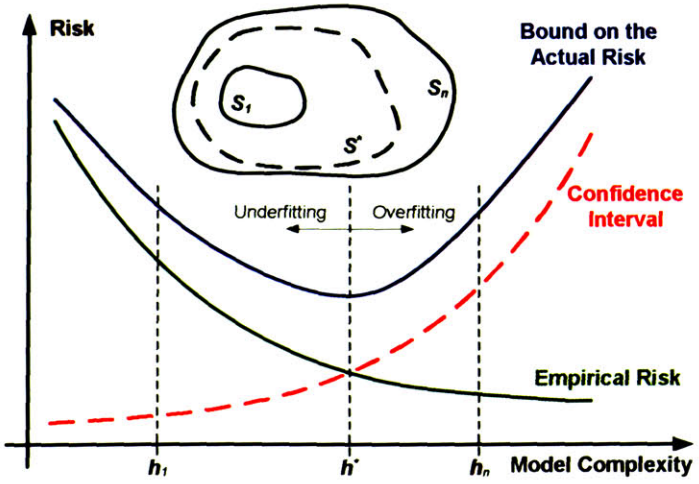


Figure 2.13 Bound on the Actual Risk (True Error)

One frequently considers a sequence of hypothesis spaces of increasing complexity and attempts to find a suitable hypothesis from each, for example neural networks with increasing numbers of hidden units. As the complexity increases the number of training errors will usually decrease, but the risk of overfitting the data correspondingly increases. By applying the theorem to each of the hypothesis spaces, we can choose the hypothesis for which the error bound is tightest.

We will see in chapter 3 that support vector machine is in fact a linear classifier in a potentially high dimensional space. It can be proved that a hyperplane in n-dimensional space has a VC dimension equal to $n+1$ (recall the special case shown in figure 2.7 ~ 2.9), and a hyperplane in high dimensional space corresponds to a large VC dimension, i.e., a complex hypothesis space.

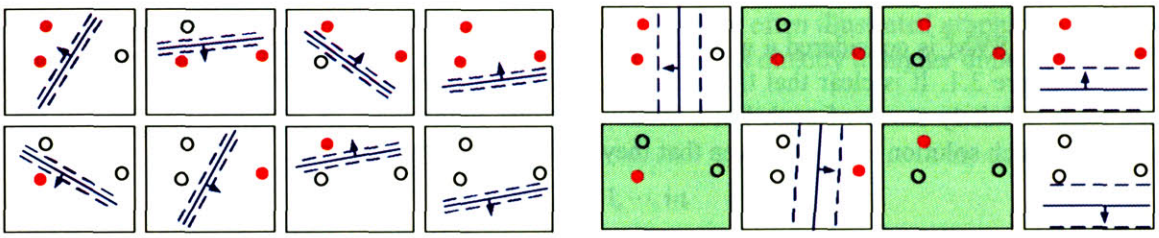


Figure 2.14 Margin and VC Dimension

Support vector machine is based on SRM. Mapping training samples into higher dimensional space corresponds to jumping from a smaller hypothesis space to a larger one, and maximize margin is a way of regularization, which compensates the complexity, as illustrated in figure 2.14. We will discuss properties of support vector machine in more detail in chapter 3.

3. Support Vector Machine

Support Vector Machine was developed by Vapnik [8] based on the structural risk minimization (SRM) principle from statistical learning theory. As we discussed in chapter two, this allow SVM to search in a rich hypothesis space without overfitting. Also, because SVM has clear mathematical background and good performance on both speed and accuracy, it has outperformed artificial neural networks (ANN) in a wide variety of fields [10]. In this chapter, I introduce and discuss the concepts underlying SVM.

3.1 Linear Support Vector Machine

Traditionally, SVM is considered a maximum margin classifier. Consider the classification problem shown in figure 3.1. It is clear that the two groups of sample points can be separated by a line, and there exist infinitely many of such lines. Without further information or assumptions, we can not determine which solution is better, given that they all have zero training error.

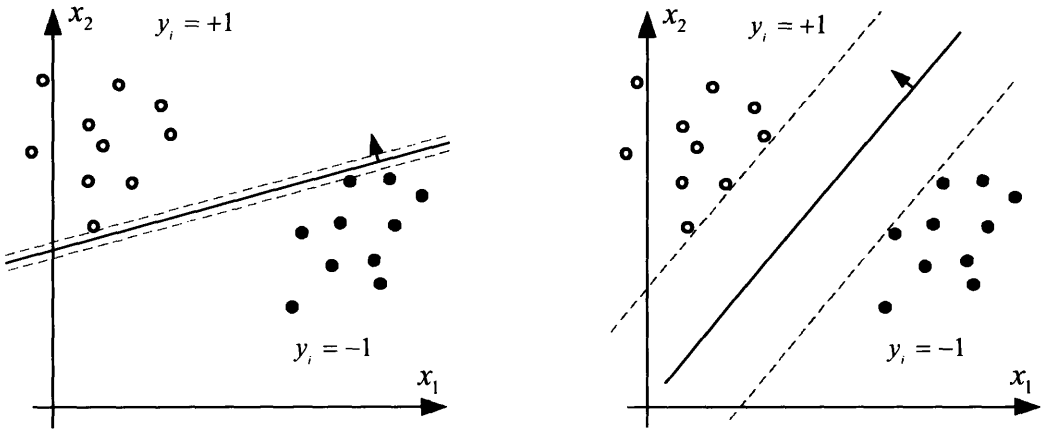


Figure 3.1. Maximal margin classifier

This is similar to an ill-conditioned system of linear equations,

$$\mathbf{A}\underset{\sim}{x} = \underset{\sim}{b} \tag{3.1}$$

or we simply have more x than the number of equations (over-determined). A unique solution can not be obtained unless we impose additional constraints, for instance, turn equation (3.1) into a least-square minimization problem and require that the Euclidean norm to be small ($\alpha > 0$),

$$\arg\min_{x} \left\|\mathbf{A}\underset{\sim}{x} - \underset{\sim}{b}\right\|^2 + \alpha\left\|\underset{\sim}{x}\right\|^2 \tag{3.2}$$

This is known as Tikhonov regularization, which improves the condition of the problem and a closed-form solution can be found,

$$\underset{\sim}{x} = \left(\mathbf{A}^T\mathbf{A} + \alpha\mathbf{I}\right)^{-1}\mathbf{A}^T\underset{\sim}{b} \tag{3.3}$$

Informally speaking, regularization is a mathematical approach of imposing stability on an ill-posed problem in a manner that yields accurate approximate solutions. The imposed constraints usually come from our prior knowledge about the problem. In classification context, the additional constraint imposed by SVM is to maximize the margin, as depicted in figure 3.1. Figure 3.1 shows a linear separable classification problem in two-dimensional space; however, SVM can be applied to nonlinear, non-separable problems in high dimensional space. To extend form linear to nonlinear, SVM first maps data from its original sample space into a high dimensional space we will refer to as feature space hereafter, and then find a hyperplane in the feature space. Non-separable problem can be dealt by introducing additional slack variables. To derive the algorithm, first consider the simplest case, a linear classifier trained on separable data. This will be extended to nonlinear, non-separable problems later on. Also, to make the concepts concrete, they are often illustrated graphically in two-dimensional space but the algorithm is general and can be applied directly to higher dimensions.

Assume we have m training points,

$$\{\underset{\sim}{x}_i, y_i\},\ i = 1, \cdots, m$$

$$\text{where } y_i \in \{-1, 1\} \text{ and } \underset{\sim}{x}_i \in \mathrm{R}^n \tag{3.4}$$

A linear classifier (a hyperplane in R^n) can be expressed as

$$f(\underset{\sim}{x}_i) = \underset{\sim}{x}_i^T\underset{\sim}{w}_1 + w_0 = 0 \tag{3.5}$$

where $\underset{\sim}{w}_1$ is a vector normal to the hyperplane, and w_0 is a scalar constant. The distance from the origin to the hyperplane is the projection of $\underset{\sim}{x}_i$ on the unit vector,

$$\underset{\sim}{x}_i^T \cdot \frac{\underset{\sim}{w}_1}{\left\|\underset{\sim}{w}_1\right\|} = \frac{-w_0}{\left\|\underset{\sim}{w}_1\right\|} \tag{3.6}$$

If we define another two parallel hyperplanes with $f(\underset{\sim}{x}_i) = \underset{\sim}{x}_i^T\underset{\sim}{w}_1 + w_0 = \pm 1$ as shown in figure 3.2, we can see that each of the two newly defined hyperplanes has a perpendicular distance to the original hyperplane equal to $1/\left\|\underset{\sim}{w}_1\right\|$. The distance is known as the "margin" between a solution hyperplane and the closest training samples. For linear separable case, SVM algorithm simply looks for the solution with maximum margin (minimum $\left\|\underset{\sim}{w}_1\right\|$) from all consistent solutions, i.e., hyperplanes that correctly separate all training samples.

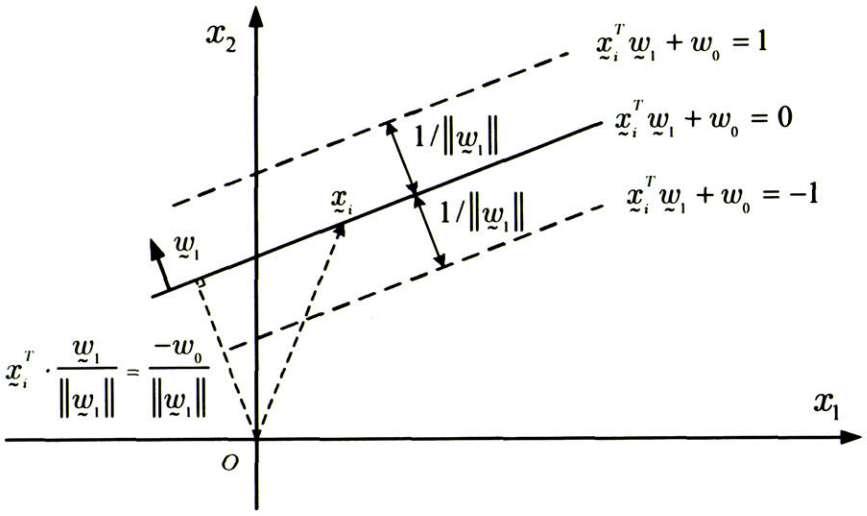


Figure 3.2. Hyperplane and margins in 2-dimensional space

The optimization problem for the above problem can be summarized as

$$\text{minimize } \frac{1}{2}\|\underset{\sim}{w}_1\|^2$$

$$\text{subject to } y_i\left(\underset{\sim}{x}_i^T \underset{\sim}{w}_1 + w_0\right) \geq 1 \tag{3.7}$$

For inseparable cases, we introduce slack variables to relax the constraints, as shown in equation (3.8) and figure 3.3.

$$\xi_i = \left(1 - y_i\left(\underset{\sim}{x}_i^T \underset{\sim}{w}_1 + w_0\right)\right)^+$$

$$\text{where } (z)^+ = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

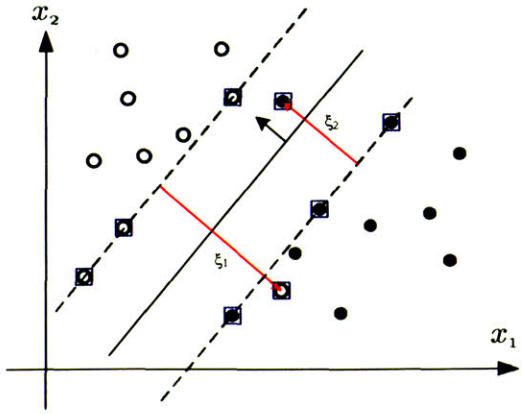


Figure 3.3. Slack Variables in a 2D classification problem

and then add some penalty to the relaxation. The new optimization problem becomes,

$$\text{minimize } \frac{1}{2}\|\underset{\sim}{w}_1\|^2 + C\sum_{i=1}^{m}\xi_i$$

$$\text{subject to } y_i\left(\underset{\sim}{x}_i^T\underset{\sim}{w}_1 + w_0\right) \geq 1-\xi_i \text{ and } \xi_i \geq 0 \tag{3.9}$$

C is a free variable used to adjust the trade-off between the regularization term and the training error penalty. Equation (3.9) reduces to equation (3.7) when C equals to infinity. We always use (3.9) instead of (3.7) because we usually do not know whether a problem is separable of not. Equation (3.9) is preferred even the problem is separable. Without the flexibility introduced by slack variables, the solution is very sensitive to noises and a small amount of outliers may alter the result dramatically, as illustrated in figure 3.4.

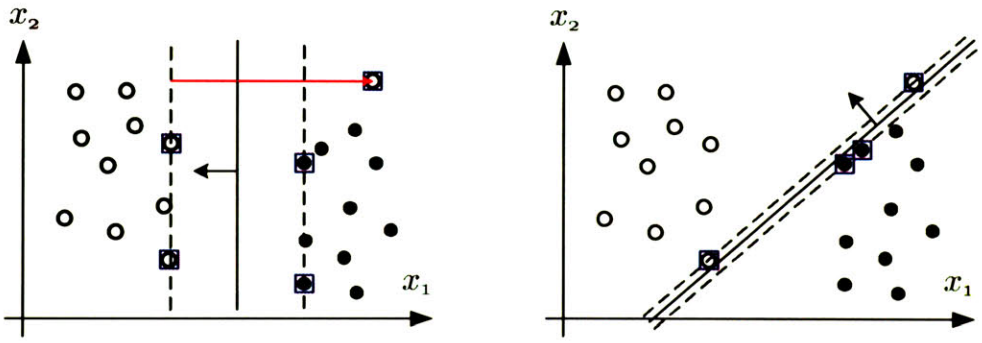


Figure 3.4. The effect of slack variable in separable problem

Equation (3.9) is an optimization problem with inequality constraints. To solve this problem, we first express constraints as losses,

$$\text{constraint}_i : c\left(\underset{\sim}{x}_i\right) \leq 0 \quad \Rightarrow \quad \text{loss}_i \equiv \max_{\alpha_i \geq 0} \alpha_i c\left(\underset{\sim}{x}_i\right) \tag{3.10}$$

Since $\alpha_i \geq 0$, if a sample does not violate the constraint, i.e., $c\left(\underset{\sim}{x}_i\right) \leq 0$, α_i must be zero in order to maximize $\alpha_i c\left(\underset{\sim}{x}_i\right)$, and $loss_i$ is therefore zero. On the other hand, if a sample violate the constraint, i.e., $c\left(\underset{\sim}{x}_i\right) > 0$, both α_i and $loss_i$ will be infinity. Also, the constraint is an active constraint when $c\left(\underset{\sim}{x}_i\right) = 0$, whereas when $c\left(\underset{\sim}{x}_i\right) < 0$, the constraint is inactive. Note that if a sample falls on the boundary, i.e., the constraint is active, then α_i can be any non-negative number.

The relation is summarized in the following,

$$
\begin{aligned}
c(\underset{\sim}{x}_i) < 0 \quad &\Rightarrow \quad loss_i = \max_{\alpha_i \geq 0} \alpha_i(-) = 0, \ \alpha_i = 0 \\
c(\underset{\sim}{x}_i) = 0 \quad &\Rightarrow \quad loss_i = \max_{\alpha_i \geq 0} \alpha_i(0) = 0, \ \alpha_i \text{ can be any non-negative number} \\
c(\underset{\sim}{x}_i) > 0 \quad &\Rightarrow \quad loss_i = \max_{\alpha_i \geq 0} \alpha_i(+) = \infty, \ \alpha_i = \infty
\end{aligned} \tag{3.11}
$$

Using this transformation, the two constraints $y_i\left(\underset{\sim}{x}_i^T \underset{\sim}{w}_1 + w_0\right) \geq 1 - \xi_i$ and $\xi_i \geq 0$ in (3.9) can be expressed as,

$$
loss_i = \max_{\alpha_i \geq 0, \beta_i \geq 0} \left\{ \alpha_i \left[1 - \xi_i - y_i\left(\underset{\sim}{x}_i^T \underset{\sim}{w}_1 + w_0\right) \right] - \beta_i \xi_i \right\} \tag{3.12}
$$

where α_i and β_i are referred to as Lagrange multipliers.

And the optimization problem becomes,

$$
\min_{\underset{\sim}{w}_1} \left\{ \frac{1}{2} \left\| \underset{\sim}{w}_1 \right\|^2 + C \sum_{i=1}^{m} \xi_i + \max_{\alpha_i \geq 0, \beta_i \geq 0} \sum_{i=1}^{m} \left\{ \alpha_i \left[1 - \xi_i - y_i\left(\underset{\sim}{x}_i^T \underset{\sim}{w}_1 + w_0\right) \right] - \beta_i \xi_i \right\} \right\} \tag{3.13}
$$

Because the first two terms do not depend on α_i and β_i, we can bring maximize to the front,

$$
\min_{\underset{\sim}{w}_1} \max_{\alpha_i \geq 0, \beta_i \geq 0} \left\{ \frac{1}{2} \left\| \underset{\sim}{w}_1 \right\|^2 + C \sum_{i=1}^{m} \xi_i + \sum_{i=1}^{m} \left\{ \alpha_i \left[1 - \xi_i - y_i\left(\underset{\sim}{x}_i^T \underset{\sim}{w}_1 + w_0\right) \right] - \beta_i \xi_i \right\} \right\} \tag{3.14}
$$

To simplify the expression, we define a new function J as below,

$$
J\left(\underset{\sim}{w}, \xi_i; \alpha_i, \beta_i\right) = \left\{ \frac{1}{2} \left\| \underset{\sim}{w}_1 \right\|^2 + C \sum_{i=1}^{m} \xi_i + \sum_{i=1}^{m} \left\{ \alpha_i \left[1 - \xi_i - y_i\left(\underset{\sim}{x}_i^T \underset{\sim}{w}_1 + w_0\right) \right] - \beta_i \xi_i \right\} \right\} \tag{3.15}
$$

where $\underset{\sim}{w}$ includes both $\underset{\sim}{w}_1$ and w_0. Using (3.15) we can rewrite (3.14) as below,

$$
\min_{\underset{\sim}{w}} \max_{\alpha_i \geq 0, \beta_i \geq 0} J\left(\underset{\sim}{w}, \xi_i; \alpha_i, \beta_i\right) = \max_{\alpha_i \geq 0, \beta_i \geq 0} \min_{\underset{\sim}{w}} J\left(\underset{\sim}{w}, \xi_i; \alpha_i, \beta_i\right) \tag{3.16}
$$

In mathematically programming problems, the left-hand-side in (3.16) is called the primal, and the right-hand-side is called the dual. Note that the solution of the primal problem and the dual problem is the same only if they satisfy the conditions of *strong duality*. For problems which do not satisfy those conditions, the primal problem is always greater than the dual problem, because we are minimizing the primal and maximizing the dual. This is known as the *weak duality*, which is always true for both convex and non-convex problems. Strong duality usually holds for a convex problem. To be more specific, a convex problem that is strictly feasible (Slater's condition [11]). The difference between the solution of primal and dual is called the *duality gap*. Because SVM is a convex quadratic programming problem, and the constraints in the form of (3.11) satisfy the conditions of strong duality, the solution of the primal is a saddle point, i.e., the duality gap is zero

and (3.16) holds. I will discuss the saddle point condition and optimization problem in more detail in next section.

Since J is differentiable and the optimal is a saddle point, we can minimize J by setting the derivatives to zero,

$$\min_{\underset{\sim}{w}} J(\underset{\sim}{w}, \xi_i; \alpha_i, \beta_i) \Rightarrow \frac{\partial J}{\partial \underset{\sim}{w}_1} = 0, \ \frac{\partial J}{\partial w_0} = 0, \text{ and } \frac{\partial J}{\partial \xi_i} = 0 \tag{3.17}$$

and

$$\frac{\partial J}{\partial \underset{\sim}{w}_1} = 0 \Rightarrow \underset{\sim}{w}_1 - \sum_{i=1}^{m} \alpha_i y_i \underset{\sim}{x}_i = 0 \tag{3.18}$$

$$\frac{\partial J}{\partial w_0} = 0 \Rightarrow \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{3.19}$$

$$\frac{\partial J}{\partial \xi_i} = 0 \Rightarrow C - \alpha_i - \beta_i = 0 \tag{3.20}$$

Substitute (3.18), (3.19), and (3.20) back into $\min_{\underset{\sim}{w}} J(\underset{\sim}{w}, \xi_i; \alpha_i, \beta_i)$, and we can rewrite the dual (the right-hand-side of (3.16)) form of the optimization problem as below,

$$\begin{aligned} &\text{maximize } \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \underset{\sim}{x}_i^T \underset{\sim}{x}_j \\ &\text{subject to } \sum_{i=1}^{m} \alpha_i y_i = 0 \text{ and } C \geq \alpha_i \geq 0 \end{aligned} \tag{3.21}$$

Recall that the discriminative function, i.e., the solution of (3.21) is

$$f(\underset{\sim}{x}_i) = \underset{\sim}{x}_i^T \underset{\sim}{w}_1 + w_0 \tag{3.22}$$

Substitute (3.18) into (3.22) we have

$$f(\underset{\sim}{x}_i) = \underset{\sim}{x}_i^T \underset{\sim}{w}_1 + w_0 = \sum_{j=1}^{m} \alpha_j y_j \underset{\sim}{x}_i^T \underset{\sim}{x}_j + w_0 \tag{3.23}$$

Note that by introducing slack variable $\xi_i \geq 0$, the constraints in (3.9), $y_i(\underset{\sim}{x}_i^T \underset{\sim}{w}_1 + w_0) \geq 1 - \xi_i$, will never be violated. For training samples fall on or outside the boundary, a simply assign a positive slack variable to the sample and make the constraint active, i.e., $y_i(\underset{\sim}{x}_i^T \underset{\sim}{w}_1 + w_0) = 1 - \xi_i$, and the relaxation will be penalized. As a result, $\underset{\sim}{x}_i$ is always feasible when using the loss function with slack variables, and an α_i can have nonzero value only when its corresponding constraint is active,

i.e., $c(\underset{\sim}{x}_i) = 0$ in (3.11). Training samples with nonzero α_i are called *support vectors*. Therefore, the solution becomes,

$$f(\underset{\sim}{x}_i) = \sum_{j=1}^{N_{SV}} \alpha_j y_j \underset{\sim}{x}_i^T \underset{\sim}{x}_j + w_0 \tag{3.24}$$

where N_{SV} is the number of support vectors. Note that those support vectors alone are sufficient to define the discriminative function, in other words, we can ignore all samples that are not support vectors from the training data set and arrive exactly the same result. For problems with a lot of noise N_{SV} can get very large and close to the total number of train samples m. However, often $N_{SV} << m$ that makes evaluating (3.24) very efficient.

Also, by definition, $\xi_i \geq 0$ when $\underset{\sim}{x}_i$ is a support vector, and similar to (3.11), the corresponding Lagrange multiplier

$$\begin{aligned} \xi_i > 0 &\Rightarrow \beta_i = 0 \\ \xi_i = 0 &\Rightarrow \beta_i \geq 0 \end{aligned} \tag{3.25}$$

Together with (3.20), we know if $\underset{\sim}{x}_i$ is a support vector,

$$\begin{aligned} \xi_i > 0 &\Rightarrow \beta_i = 0 \Leftrightarrow \alpha_i = C \\ \xi_i = 0 &\Rightarrow \beta_i \geq 0 \Leftrightarrow C \geq \alpha_i \geq 0 \end{aligned} \tag{3.26}$$

and remember $\alpha_i = 0$ if $\underset{\sim}{x}_i$ is not a support vector.

As can be seen in (3.21), introducing the dual representation results in a simpler optimization problem than its primal form. Further, a more important reason is that in the dual form, the sample vectors only appear in the dot product form, which is a very important property of SVM and I will discuss more in section 3.4 when I discuss the properties of kernels.

3.2 Karush-Kuhn-Tucker Conditions for Differentiable Convex Problem

The technique we used when solving (3.9) can be thought of as a form of a generalized Lagrange multipliers, and is a standard procedure when dealing with nonlinear optimization problem. In a general nonlinear optimization problem, the necessary and sufficient conditions of an optimal are referred to as the Karush-Kuhn-Tucker (KKT) conditions, also known as the Kuhn-Tucker conditions. This is a fundamental and well known property in optimization, and is also of great importance in SVM. Because we use the KKT conditions to transform the primal optimization problem into its dual form and solve the dual numerically, we must ensure that the solvability of the optimization problem does not change after the transformation. The following discussion of KKT follows [12] and [13]. A general nonlinear optimization problem with both equality and inequality constraints is,

$$\text{Minimize } f(\underset{\sim}{x})$$

$$\text{Subject to } c_i^{\text{in}}(\underset{\sim}{x}) \le 0 \quad \text{and} \quad c_j^{\text{eq}}(\underset{\sim}{x}) = 0 \tag{3.27}$$

the corresponding Lagrangian can be defined as,

$$L(\underset{\sim}{x}, \underset{\sim}{\alpha}, \underset{\sim}{\beta}) \equiv f(\underset{\sim}{x}) + \sum_{i=1}^{n} \alpha_i c_i^{\text{in}}(\underset{\sim}{x}) + \sum_{j=1}^{m} \beta_j c_j^{\text{eq}}(\underset{\sim}{x}) \quad \text{where} \quad \alpha_i \ge 0 \quad \text{and} \quad \beta_j \ge 0 \tag{3.28}$$

The α_i and β_j are called the Lagrange multiplier. For simplicity, we remove the equality constraints from the following discussion but they are simpler and can be easily extended once we know how to deal with inequality constraints. Consider only inequality constraints, (3.28) can be simplified as,

$$L(\underset{\sim}{x}, \underset{\sim}{\alpha}) \equiv f(\underset{\sim}{x}) + \sum_{i=1}^{n} \alpha_i c_i(\underset{\sim}{x}) \quad \text{where} \quad \alpha_i \ge 0 \tag{3.29}$$

If there exists a pair of variables $(\underset{\sim}{x}^*, \underset{\sim}{\alpha}^*)$ such that for all $\underset{\sim}{x}$ and $\underset{\sim}{\alpha}$,

$$L(\underset{\sim}{x}^*, \underset{\sim}{\alpha}) \le L(\underset{\sim}{x}^*, \underset{\sim}{\alpha}^*) \tag{3.30}$$

$$L(\underset{\sim}{x}^*, \underset{\sim}{\alpha}^*) \le L(\underset{\sim}{x}, \underset{\sim}{\alpha}^*) \tag{3.31}$$

$(\underset{\sim}{x}^*, \underset{\sim}{\alpha}^*)$ is called the saddle point and $\underset{\sim}{x}^*$ is a solution to (3.27). Substituting (3.29) into (3.30) we have,

$$\sum_{i=1}^{n} (\alpha_i - \alpha_i^*) c_i(\underset{\sim}{x}^*) \le 0 \tag{3.32}$$

Because the only constraint on $\underset{\sim}{\alpha}$ is $\alpha_i \ge 0$, we can set it to be any nonzero number and both (3.30) and (3.31) still hold. By choosing $\alpha_i = \alpha_i^* + 1$ and $\alpha_1 = \alpha_1^*$, ..., $\alpha_{i-1} = \alpha_{i-1}^*$, $\alpha_{i+1} = \alpha_{i+1}^*$, ..., $\alpha_n = \alpha_n^*$, we can see that

$$c_i(\underset{\sim}{x}^*) \le 0 \text{ for all } i \tag{3.33}$$

i.e., $\underset{\sim}{x}^*$ satisfies all constraints and this it is feasible. Similarly, by choosing $\alpha_i = 0$ and $\alpha_1 = \alpha_1^*$, ..., $\alpha_{i-1} = \alpha_{i-1}^*$, $\alpha_{i+1} = \alpha_{i+1}^*$, ..., $\alpha_n = \alpha_n^*$, we have

$$\alpha_i^* c_i(\underset{\sim}{x}^*) \ge 0 \text{ for all } i \tag{3.34}$$

Since $\alpha_i^* \geq 0$ and $c_i\left(\underset{\sim}{x}^*\right) \leq 0$ for all i, to keep (3.34) valid the only possible combinations are,

	$c_i\left(\underset{\sim}{x}^*\right)=0$	$c_i\left(\underset{\sim}{x}^*\right)<0$
$\alpha_i^*=0$	$\alpha_i^* c_i\left(\underset{\sim}{x}^*\right)=0$	$\alpha_i^* c_i\left(\underset{\sim}{x}^*\right)=0$
$\alpha_i^*>0$	$\alpha_i^* c_i\left(\underset{\sim}{x}^*\right)=0$	Not Valid

that can be summarized as,

$$\alpha_i^* c_i\left(\underset{\sim}{x}^*\right)=0 \text{ for all } i \tag{3.35}$$

Note that α_i^* have nonzero value only when $c_i\left(\underset{\sim}{x}^*\right)=0$. Equation (3.35) is often referred to as the *complementary slackness* condition. To make the concept concrete, (3.35) is illustrated graphically in figures 3.5.

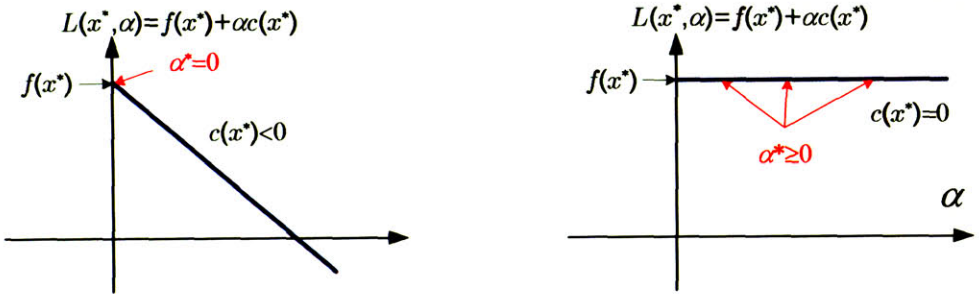


Figure 3.5 Complementary Slackness and Saddle Point Condition

Here we assume $\underset{\sim}{x}$ is a scalar, i.e., $\underset{\sim}{x}=x$, and there is only one inequality constraint $c_i(x) \leq 0$. Given $x=x^*$, $c_i\left(x^*\right)$ is an non-positive constant and the Lagrange $L\left(x,\alpha \mid x=x^*\right)$ is a linear function with respect to α. Since (3.30) requires $L\left(x^*,\alpha\right) \leq L\left(x^*,\alpha^*\right)$, i.e., $\alpha_i c_i\left(\underset{\sim}{x}^*\right) \leq \alpha_i^* c_i\left(\underset{\sim}{x}^*\right)$ for all α_i, α_i^* must be zero when $c_i\left(\underset{\sim}{x}^*\right)$ is not zero.

Note that because α is constrained (non-negative), the derivative of $L\left(x,\alpha \mid x=x^*\right)$ with respect to α is not necessary zero, as shown in the left hand side of figure 3.5.

Finally, substituting (3.29) into (3.31) yields,

$$f\left(\underset{\sim}{x}^*\right)-f\left(\underset{\sim}{x}\right)+\sum_{i=1}^{n}\alpha_i^* c_i\left(\underset{\sim}{x}^*\right)-\sum_{i=1}^{n}\alpha_i^* c_i\left(\underset{\sim}{x}\right) \leq 0 \tag{3.36}$$

Because $\alpha_i^* c_i(\underset{\sim}{x}^*) = 0$ and $-\alpha_i^* c_i(\underset{\sim}{x}) \geq 0$, we have $f(\underset{\sim}{x}^*) \leq f(\underset{\sim}{x})$ for all feasible $\underset{\sim}{x}$, regardless the value of $\underset{\sim}{\alpha}^*$. This prove that the saddle point, (3.30) and (3.31), is a *sufficient* condition for $\underset{\sim}{x}^*$ to be a global minimum, i.e., a solution of (3.27). Again, considering the case in figure 3.5 and assume $f(x)$ is convex and differentiable, we can illustrate (3.31) as shown in figure 3.6,

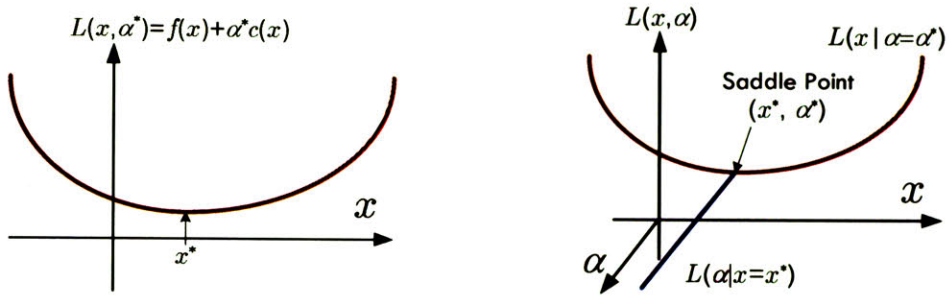


Figure 3.6 Saddle Point Condition for Convex, Differentiable Objective Function

Notice that because x is not constrained, the derivative of $L(x, \alpha \mid \alpha = \alpha^*)$ with respect to x must vanish at the optimal.

The saddle point condition is also a *necessary* condition for $\underset{\sim}{x}^*$ to be an optimal, provided that the feasible region **X** is a convex set and both $f(\underset{\sim}{x})$ and $c_i(\underset{\sim}{x})$ are convex functions on X, and $c_i(\underset{\sim}{x})$ satisfies one of the following constraint qualifications,

1. The feasible region **X** contains at least two distinct elements, and there exists an $\underset{\sim}{x} \in \mathbf{X}$ such that all $c_i(\underset{\sim}{x})$ are strictly convex at $\underset{\sim}{x}$ with respect to **X**.
2. $\exists \underset{\sim}{x} \in \mathbf{X}$ such that $c_i(\underset{\sim}{x}) < 0$ for all i (Slater's strictly feasible condition)
3. $\forall \alpha \in [0, \infty)$, $\exists \underset{\sim}{x} \in \mathbf{X}$ such that $\sum_i \alpha_i c_i(\underset{\sim}{x}) \leq 0$

The prove is more lengthy and can be found in [12]. However, put this into SVM context, we can see that the optimization problem satisfies all the conditions and therefore the saddle point condition is a necessary and sufficient condition for an optimal solution in SVM. Further, for a SVM problem, both $f(\underset{\sim}{x})$ and $c_i(\underset{\sim}{x})$ are differentiable, and the saddle point conditions can be rewritten as following,

$$c_i^{\text{in}}(\underset{\sim}{x}^*) \leq 0 \quad \text{(Inequality Primal Constraints)} \tag{3.37}$$

$$c_j^{\text{eq}}(\underset{\sim}{x}^*) = 0 \quad \text{(Equality Primal Constraints)} \tag{3.38}$$

$$\alpha_i^* \geq 0 \quad \text{(Dual Constraints)} \tag{3.39}$$

$$\alpha_i^* c_i^{in}(\underset{\sim}{x}^*) = 0 \text{ for all } i \quad \text{(Complementary Slackness)} \tag{3.40}$$

$$\nabla_{\underset{\sim}{x}}\left[f(\underset{\sim}{x}^*) + \sum_{i=1}^{n} \alpha_i^* c_i^{in}(\underset{\sim}{x}^*) + \sum_{j=1}^{m} \beta_j^* c_j^{eq}(\underset{\sim}{x}^*) \right] = 0 \quad \text{(Saddle Point in } \underset{\sim}{x}^*) \tag{3.41}$$

Because the KKT conditions are both sufficient and necessary, solving (3.37) to (3.41) is equivalent to solving the original optimization problem. After transforming an optimization problem into solving a set of equations, it can be solved by many existing numerical algorithms efficiently, for instance, interior point method. I will discuss this issue in more detail in the implementation chapter.

3.3 Nonlinear Support Vector Machine

To extend the algorithm to nonlinear cases, we can define a mapping function $\underset{\sim}{\phi}: \mathrm{R}^n \mapsto \mathcal{H}$ that map $\underset{\sim}{x}_i$ from its original Euclidian space to a reproducing kernel Hilbert space (RKHS). Without losing generality in our context, we can simply think a RKHS as a generalization Euclidian space that can have infinite dimension. Then we can replace $\underset{\sim}{x}_i$ in the optimization problem with $\underset{\sim}{\phi}(\underset{\sim}{x}_i)$ and perform linear classification in $\mathcal{H}$. Finding a hyperplane in $\mathcal{H}$ is no longer a linear operation in the original Euclidian space. We will call the original Euclidian space the *sample space*, and $\mathcal{H}$ will be referred to as the *feature space* hereafter.

Consider the one-dimensional classification problem shown in figure 3.6. It is clear that no single line (x = constant) can separate the starts from the disks in the 1D space. However, if we choose a mapping function $\underset{\sim}{\phi}: x_{1i} \mapsto \left\{x_{1i}, (x_{1i}-5)^2\right\}$ that maps the original samples to 2D, and the samples are separable in the transformed space.

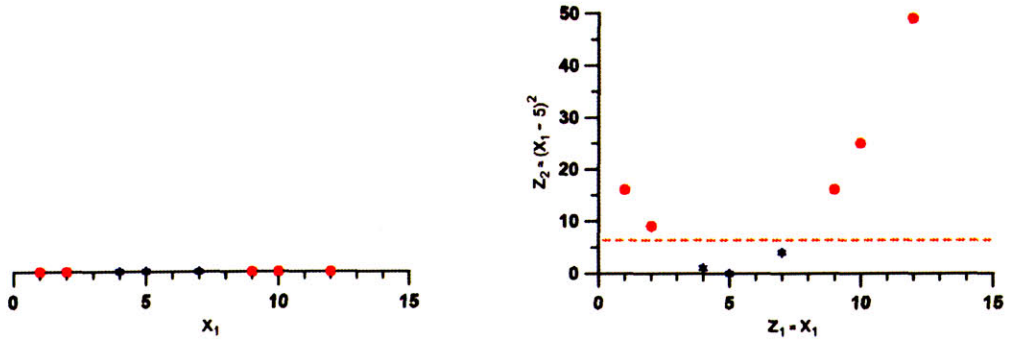


Figure 3.6. A linear inseparable problem in 1D is linear separable in 2D through proper nonlinear mapping

A slightly more complicated example is shown in figure 3.7, where the samples in the original space is not linear separable and needs a quadratic curve, for instance, a ellipse, is needed. Because a ellipse can be expressed as $x_1^2/a^2 + x_2^2/b^2 = 1$, it can be thought of as a line in the $\left\{x_1^2, x_2^2\right\}$ space. A hyperplane in any space that contains x_1^2 and x_2^2 is sufficient to separate samples inside the ellipse from samples outside the ellipse. The idea is illustrated in Figure 3.7, where a mapping

function transforms the samples from the original sample space to a space of monomial of degree 2, $\underset{\sim}{\phi}: \{x_{1i}, x_{2i}\} \mapsto \{x_{1i}^2, x_{1i}x_{2i}, x_{2i}^2\}$ is selected.

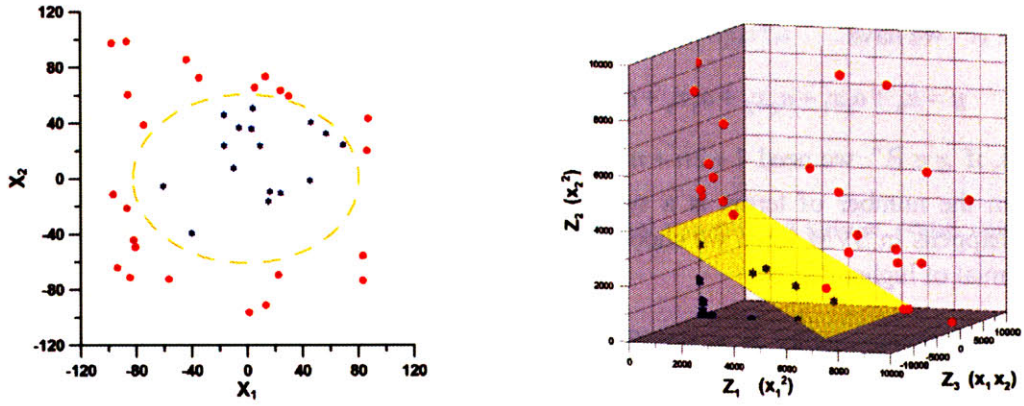


Figure 3.7. An linear inseparable problem in 2D is linear separable in 3D through proper nonlinear mapping

As shown in the previous examples, a linear inseparable problem might be separated by a hyperplane in a higher dimension. It is clear that the higher the dimension is, the easier a hyperplane can separate the data. From the VC-dimension point of view, a hyperplane in more complex space has the capability of shattering more points. This is also intuitive because the data points become much sparser in high dimension space. However, a higher dimension is not always necessary, as discussed in the ellipse example. Some kernel functions, such as radial basis functions, are corresponded to an infinite dimensional space and thus can shatter any number of training data. This gives us zero training error but it does not guarantee good generalization. We will get back to this point in more detail in the later section.

3.4 Kernel in Support Vector Machine

3.4.1 The Curse of Dimensionality

When training a SVM using (3.21), the training samples need to be mapped into the selected feature space by a mapping function $\underset{\sim}{\phi}: \mathrm{R}^n \mapsto \mathcal{H}$, and the dot product of all pairs of the samples $\underset{\sim}{\phi}(\underset{\sim}{x}_i)^T \underset{\sim}{\phi}(\underset{\sim}{x}_j)$ need to be calculated. Because feature space is often very high dimensional, performing dot product for all pairs of training samples become computationally intractable problem for any nontrivial problem. The difficulties occur as the dimensionality of a problem is increased is referred to as the *curse of dimensionality*, and it imposes serious limitation to many of the learning algorithms that give impressive results for low dimensional problems. To make this idea concrete, let us say that we wanted to do a second-order polynomial regression. In one-dimension $x \in \mathrm{R}^1$ the function is simply

$$y = a_0 + a_1 x + a_2 x^2 \tag{3.42}$$

However, the length of the terms increases quickly, in $\underset{\sim}{x} \in \mathrm{R}^2$,

$$y = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2 \tag{3.43}$$

and in $\underset{\sim}{x} \in \mathrm{R}^3$ we have

$$y = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_3 + a_4 x_1 x_2 + a_5 x_1 x_3 + a_6 x_2 x_3 + a_7 x_1^2 + a_8 x_2^2 + a_9 x_3^2 \tag{3.44}$$

In general, if $\underset{\sim}{x} \in \mathrm{R}^n$ we need $1 + 2n + n(n-1)/2$ terms for a second-order polynomial. The rapid increase in the number of terms is a typical example of the curse of dimensionality. The same situation happens in SVM when we choose to map training samples to a feature space of monomial or polynomial of higher degree.

The curse of dimensionality is a significant obstacle in machine learning problems that involve learning from data samples in a high-dimensional feature space. Fortunately, SVM can get around the curse of dimensionality by using a technique sometimes known as the *kernel trick*. Note that in the dual representation of SVM, only the dot products of the training samples, $\underset{\sim}{x}_i^T \underset{\sim}{x}_j$, are needed in both training (3.21) and testing (3.24) stages. Similarly, only $\underset{\sim}{\phi}(\underset{\sim}{x}_i)^T \underset{\sim}{\phi}(\underset{\sim}{x}_j)$ is needed for a nonlinear SVM. If there exists a function that satisfies,

$$\mathcal{K}(\underset{\sim}{x}_i, \underset{\sim}{x}_j) = \underset{\sim}{\phi}(\underset{\sim}{x}_i)^T \underset{\sim}{\phi}(\underset{\sim}{x}_j) \tag{3.45}$$

and $\mathcal{K}(\underset{\sim}{x}_i, \underset{\sim}{x}_j)$ can be evaluated efficiently, we can then use $\mathcal{K}(\underset{\sim}{x}_i, \underset{\sim}{x}_j)$ instead of computing the mapping $\underset{\sim}{x}_i \mapsto \underset{\sim}{\phi}(\underset{\sim}{x}_i)$ and the dot product in feature space explicitly. The function $\mathcal{K}(\underset{\sim}{x}_i, \underset{\sim}{x}_j)$ is known as the *kernel function*. Take a second-order monomial mapping for example, the mapping is

$$\underset{\sim}{x}_i = \begin{Bmatrix} x_{i1} \\ x_{i2} \end{Bmatrix} \quad \mapsto \quad \underset{\sim}{\phi}(\underset{\sim}{x}_i) = \begin{Bmatrix} x_{i1}^2 \\ x_{i2}^2 \\ x_{i1} x_{i2} \\ x_{i2} x_{i1} \end{Bmatrix} \tag{3.46}$$

and the dot product is,

$$\begin{aligned} \underset{\sim}{\phi}(\underset{\sim}{x}_i)^T \underset{\sim}{\phi}(\underset{\sim}{x}_j) &= \{x_{i1}^2, x_{i2}^2, x_{i1} x_{i2}, x_{i2} x_{i1}\} \{x_{j1}^2, x_{j2}^2, x_{j1} x_{j2}, x_{j2} x_{j1}\}^T \\ &= x_{i1}^2 x_{j1}^2 + x_{i2}^2 x_{j2}^2 + x_{i1} x_{i2} x_{j1} x_{j2} + x_{i2} x_{i1} x_{j2} x_{j1} \end{aligned} \tag{3.47}$$

Both steps require much more computational power than performing dot product in the original sample space, and it is easy to see that when the degree of dimensionality increases the computation cost will eventually become intractable. However, the dot product of the monomial in feature space can be expressed as the exponentiation of the dot product in the sample space,

$$
\begin{aligned}
k(\underset{\sim}{x}_i, \underset{\sim}{x}_j) &= \underset{\sim}{\phi}(\underset{\sim}{x}_i)^T \underset{\sim}{\phi}(\underset{\sim}{x}_j) \\
&= \left\{x_{i1}^2, x_{i2}^2, x_{i1}x_{i2}, x_{i2}x_{i1}\right\}\left\{x_{j1}^2, x_{j2}^2, x_{j1}x_{j2}, x_{j2}x_{j1}\right\}^T \\
&= x_{i1}^2 x_{j1}^2 + x_{i2}^2 x_{j2}^2 + x_{i1}x_{i2}x_{j1}x_{j2} + x_{i2}x_{i1}x_{j2}x_{j1} \\
&= \left(x_{i1}x_{j1} + x_{i2}x_{j2}\right)^2 \\
&= \left(\underset{\sim}{x}_i^T \underset{\sim}{x}_j\right)^2
\end{aligned}
\tag{3.48}
$$

If we choose to use polynomial mapping instead of monomial mapping, a similar result can also be obtained and the corresponding polynomial kernel is

$$
k(\underset{\sim}{x}_i, \underset{\sim}{x}_j) = \left(r + \underset{\sim}{x}_i^T \underset{\sim}{x}_j\right)^d \tag{3.49}
$$

In general, if $\underset{\sim}{x}_i$ has n components and we choose to map the samples to a feature space consists of polynomial of degree d, the total number of terms, i.e., the dimensionality of the feature space is $C_d^{n+d} = (n+d)!/(n!d!)$. However, if (3.49) is used, the calculation is almost the same as doing dot product in the original sample space, even when the samples are mapped into a feature space with very high dimensionality.

3.4.2 Examples of Kernels

When no further information is available except a set of training samples is given, three kernels are used in SVM frequently in practice, namely, linear kernel, polynomial kernel, and radial basis kernel. The linear kernel,

$$
\mathcal{K}(\underset{\sim}{x}_i, \underset{\sim}{x}_j) = \underset{\sim}{x}_i^T \underset{\sim}{x}_j \tag{3.50}
$$

corresponding to a dot product in the sample space, and is the simplest form of a dot product kernel. When using SVM, all sample related information is transformed into a kernel, thus the kernel can be thought of as an information bottleneck in SVM. In the linear kernel case, the kernel represents pair-wise similarity in sample space among all training samples. A homogeneous polynomial kernel,

$$
\mathcal{K}(\underset{\sim}{x}_i, \underset{\sim}{x}_j) = \left(\underset{\sim}{x}_i^T \underset{\sim}{x}_j\right)^d \tag{3.51}
$$

has a similar meaning; however, we have shown that the similarity metric is now defined in a feature space whose dimensions are spanned by all possible d-th order monomials of the samples. An inhomogeneous polynomial is often used,

$$
\mathcal{K}(\underset{\sim}{x}_i, \underset{\sim}{x}_j) = \left(\underset{\sim}{x}_i^T \underset{\sim}{x}_j + c\right)^d \tag{3.52}
$$

which performs the optimization in a potentially high dimensional feature space implicitly and only requires marginal increase in computational resources, comparing to using linear kernel. Gaussian radial basis function kernel is often suggested [14],

$$\mathcal{K}\left(\underset{\sim}{x}_i, \underset{\sim}{x}_j\right) = \exp\left(\frac{-\left\|\underset{\sim}{x}_i - \underset{\sim}{x}_j\right\|^2}{2\sigma^2}\right) \tag{3.53}$$

Since Gaussian RBF satisfies Mercer's condition of semi-positive, it is in fact a dot product in some feature space. Notice that $\mathcal{K}\left(\underset{\sim}{x}_i, \underset{\sim}{x}_i\right) = e^0 = 1$, therefore $\underset{\sim}{\phi}\left(\underset{\sim}{x}_i\right)^T \underset{\sim}{\phi}\left(\underset{\sim}{x}_i\right) = \left\|\underset{\sim}{\phi}\left(\underset{\sim}{x}_i\right)\right\| = 1$, i.e., all mapped samples in feature has unit length. It can be proven [15] that a matrix $\mathbf{K}$ given by $\mathbf{K}_{ij} = \exp\left(-\left\|\underset{\sim}{x}_i - \underset{\sim}{x}_j\right\|^2 / 2\sigma^2\right)$ has full rank provided that no two $\underset{\sim}{x}_i$ are the same and $\sigma > 0$. In other words any two row vectors in the kernel matrix, e.g., $\underset{\sim}{\phi}\left(\underset{\sim}{x}_i\right)^T \left\{\underset{\sim}{\phi}\left(\underset{\sim}{x}_1\right), \cdots, \underset{\sim}{\phi}\left(\underset{\sim}{x}_m\right)\right\}$ and $\underset{\sim}{\phi}\left(\underset{\sim}{x}_j\right)^T \left\{\underset{\sim}{\phi}\left(\underset{\sim}{x}_1\right), \cdots, \underset{\sim}{\phi}\left(\underset{\sim}{x}_m\right)\right\}$, are linearly independent. Thus, all mapped sample vectors in the feature space, $\underset{\sim}{\phi}\left(\underset{\sim}{x}_1\right), \cdots, \underset{\sim}{\phi}\left(\underset{\sim}{x}_m\right)$ are also linearly independent, i.e., those samples span an m-dimensional space. Because there is no restriction on the number of training samples m, a Gaussian RBF produces a feature space with infinite dimensions.

It can be shown that any symmetric kernel function satisfying Mercer's condition corresponding to a dot product in some feature space, and in many cases, one does not need to know what the mapping function $\underset{\sim}{\phi}$ is because only the dot product that matters. Nevertheless, it is helpful in understanding the phenomenon if the mapping function is known. I will discuss this in further detail in chapter 4.

3.5 Support Vector Regression

SVM was first developed for classification, and the labels y_i in (3.4) represent a finite number of possible categories. The algorithm can be extended to estimate real-valued functions by defining a suitable loss function and allowing y_i to have real value [16]. One example of such loss function is,

$$\xi_i = \left(\left|f\left(\underset{\sim}{x}_i\right) - y_i\right| - \varepsilon\right)^+ = \max\left(\left|f\left(\underset{\sim}{x}_i\right) - y_i\right| - \varepsilon,\ 0\right) \tag{3.54}$$

Equation (3.54) is called the ε–insensitive loss function. It pays no penalty to samples within the ε–tube and linear loss to samples outside the tube, as shown in figure 3.8. A proper loss function like this carries over the sparseness property from SVM to SVR. The function of ξ_i here in regression is similar to the function of slack variables used in inseparable classification (3.8), except we need two types of slack variables in regression so we can tell which side a sample is, i.e., $y_i - f\left(\underset{\sim}{x}_i\right) > \varepsilon$ or $f\left(\underset{\sim}{x}_i\right) - y_i > \varepsilon$. Recall that in classification, only one kind of slack variable is required because the constraint, $y_i\left(\underset{\sim}{x}_i^T \underset{\sim}{w}_1 + w_0\right) \geq 1 - \xi_i$, works for both $y_i = 1$ and $y_i = -1$.

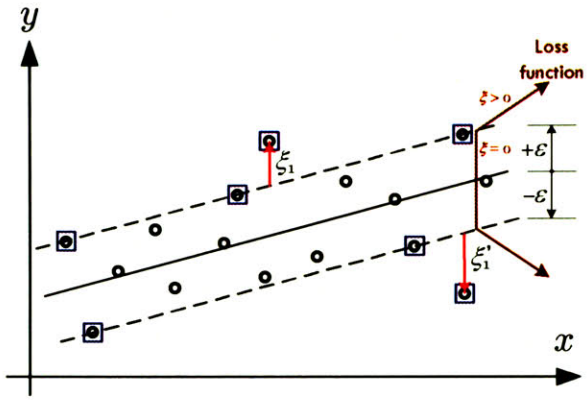


Figure 3.8 Support Vector Regression and ε–insensitive Loss Function

Similar to (3.9),

$$\text{minimize } \frac{1}{2}\|\underset{\sim}{w}_1\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i')$$

$$\text{subject to } \begin{matrix} y_i - f(\underset{\sim}{x}_i) - \varepsilon \le \xi_i \quad \text{and} \quad \xi_i \ge 0 \\ f(\underset{\sim}{x}_i) - y_i - \varepsilon \le \xi_i' \quad \text{and} \quad \xi_i' \ge 0 \end{matrix} \tag{3.55}$$

Note that for any training sample, $\xi_i + \xi_i' = \xi_i$ or $\xi_i + \xi_i' = \xi_i'$. The dual from of (3.55) can be obtained the same way we did (3.21),

$$\text{maximize } \sum_{i=1}^{m}(\alpha_i - \alpha_i')(y_i - \varepsilon) - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}(\alpha_i - \alpha_i')(\alpha_j - \alpha_j')\underset{\sim}{x}_i^T\underset{\sim}{x}_j$$

$$\text{subject to } \sum_{i=1}^{m}(\alpha_i - \alpha_i') = 0,\ C \ge \alpha_i, \text{ and } \alpha_i' \ge 0 \tag{3.56}$$

and the solution, i.e., the regression function is

$$f(\underset{\sim}{x}_i) = \underset{\sim}{x}_i^T\underset{\sim}{w}_1 + w_0 = \sum_{j=1}^{m}(\alpha_j - \alpha_j')\underset{\sim}{x}_i^T\underset{\sim}{x}_j + w_0 \tag{3.57}$$

Again, only pair-wise dot product of training samples, $\underset{\sim}{x}_i^T\underset{\sim}{x}_j$, appears in both training (3.56) and predicting (3.57). The discussion on kernel mapping can be applied directly in support vector regression.

3.6 Summary

We end this chapter by summarize the properties of SVM and give it an analogical interpretation in mechanics. First, we can rewrite the primal optimization problem for non-separable cases in (3.9) as Lagrangian,

$$L\left(\underset{\sim}{w},\underset{\sim}{x},\underset{\sim}{\xi},\underset{\sim}{\alpha},\underset{\sim}{\beta}\right)=\frac{1}{2}\left\|\underset{\sim}{w}_1\right\|^2+C\sum_{i=1}^{m}\xi_i-\sum_{i=1}^{m}\alpha_i\left[y_i\left(\underset{\sim}{x}_i^T\underset{\sim}{w}_1+w_0\right)-1+\xi_i\right]-\sum_{i=1}^{m}\beta_i\xi_i \tag{3.58}$$

and the sufficient and necessary KKT conditions for an optimal are (cf. (3.37) ~ (3.41)),

$$\frac{\partial}{\partial\alpha_i}L\left(\underset{\sim}{w},\underset{\sim}{\xi},\underset{\sim}{\alpha},\underset{\sim}{\beta}\right)=-y_i\left(\underset{\sim}{x}_i^T\underset{\sim}{w}_1+w_0\right)+1-\xi_i\le 0 \quad \text{(Primal Constraints)} \tag{3.59}$$

$$\frac{\partial}{\partial\beta_i}L\left(\underset{\sim}{w},\underset{\sim}{\xi},\underset{\sim}{\alpha},\underset{\sim}{\beta}\right)=-\xi_i\le 0 \quad \text{(Primal Constraints)} \tag{3.60}$$

$$\alpha_i\ge 0 \quad \text{(Dual Constraints)} \tag{3.61}$$

$$\beta_i\ge 0 \quad \text{(Dual Constraints)} \tag{3.62}$$

$$\alpha_i\left[y_i\left(\underset{\sim}{x}_i^T\underset{\sim}{w}_1+w_0\right)-1+\xi_i\right]=0 \quad \text{(Complementary Slackness)} \tag{3.63}$$

$$\beta_i\xi_i=0 \quad \text{(Complementary Slackness)} \tag{3.64}$$

$$\frac{\partial}{\partial w_0}L\left(\underset{\sim}{w},\underset{\sim}{\xi},\underset{\sim}{\alpha},\underset{\sim}{\beta}\right)=-\sum_{i=1}^{m}\alpha_i y_i=0 \quad \text{(Saddle Point)} \tag{3.65}$$

$$\frac{\partial}{\partial\underset{\sim}{w}_1}L\left(\underset{\sim}{w},\underset{\sim}{\xi},\underset{\sim}{\alpha},\underset{\sim}{\beta}\right)=\underset{\sim}{w}_1-\sum_{i=1}^{m}\alpha_i y_i\underset{\sim}{x}_i=0 \quad \text{(Saddle Point)} \tag{3.66}$$

$$\frac{\partial}{\partial\xi_i}L\left(\underset{\sim}{w},\underset{\sim}{\xi},\underset{\sim}{\alpha},\underset{\sim}{\beta}\right)=C-\alpha_i-\beta_i=0 \quad \text{(Saddle Point)} \tag{3.67}$$

where (3.59) and (3.60) are the primal constraints; (3.61) and (3.62) the dual constraints, also knows as the Lagrange multipliers; (3.63) and (3.64) are the complementary slackness property. Equations (3.65) ~ (3.67) are the saddle point conditions with respect to the primal variables, w_0, $\underset{\sim}{w}_1$, and ξ_i, respectively. Recall that $\alpha_i\ne 0$ only when $y_i\left(\underset{\sim}{x}_i^T\underset{\sim}{w}_1+w_0\right)-1+\xi_i=0$, i.e., the constraint is active. This can be easily seen from (3.63). Those training samples that have nonzero α_i are called the support vectors.

Recall that the geographic meaning of SVM is finding a maximum margin discrimination hyperplane that subjects to some constraints. Let us think the discrimination hyperplane together with the two margins as a plate, which always tend to expand if not constrained. The process of finding a maximum margin solution subjects to constraints can then be thought of as putting that special plate to a space with some obstacles, as shown in figure 3.9,

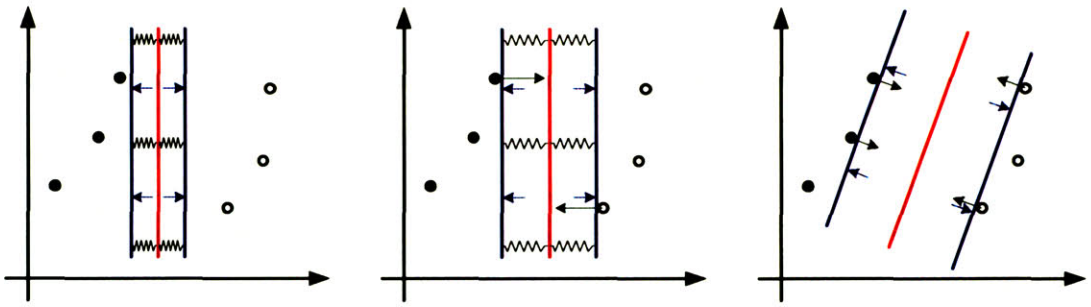


Figure 3.9 Hard-Margin Support Vector Machine

A sample point applies a reaction force to the margin plane when it is touched by the plane. As can be seen in the middle of figure 3.9, the plate starts to rotate when its boundary hits the obstacles, i.e., a support vector. The plate stops expanding when it reaches a static equilibrium state. We can think the value of α_i as the magnitude of the reaction force from each training sample $\underset{\sim}{x}_i$. It is obvious that only the support vectors have non-zero contributions. Further, each reaction force can be expressed as $\vec{F}_i = \alpha_i y_i \underset{\sim}{w}_1 / \|\underset{\sim}{w}_1\|$, where $y_i \underset{\sim}{w}_1 / \|\underset{\sim}{w}_1\|$ is a unit vector indicates the direction of the reaction force. Because $\sum_{i=1}^{m} \alpha_i y_i = 0$, we have,

$$\sum_{i=1}^{n_{SV}} \vec{F}_i = \sum_{i=1}^{n_{SV}} \alpha_i y_i \frac{\underset{\sim}{w}_1}{\|\underset{\sim}{w}_1\|} = \frac{\underset{\sim}{w}_1}{\|\underset{\sim}{w}_1\|} \sum_{i=1}^{n_{SV}} \alpha_i y_i = 0 \tag{3.68}$$

Also,

$$\sum_{i=1}^{n_{SV}} \vec{F}_i \times \underset{\sim}{x}_i = \sum_{i=1}^{n_{SV}} \left(\alpha_i y_i \frac{\underset{\sim}{w}_1}{\|\underset{\sim}{w}_1\|} \times \underset{\sim}{x}_i \right) = \sum_{i=1}^{n_{SV}} \left(\frac{\underset{\sim}{w}_1}{\|\underset{\sim}{w}_1\|} \times \alpha_i y_i \underset{\sim}{x}_i \right) = \frac{\underset{\sim}{w}_1}{\|\underset{\sim}{w}_1\|} \times \sum_{i=1}^{n_{SV}} \left(\alpha_i y_i \underset{\sim}{x}_i \right) = \frac{\underset{\sim}{w}_1}{\|\underset{\sim}{w}_1\|} \times \underset{\sim}{w}_1 = 0 \tag{3.69}$$

where operator $\times$ means cross product of vectors. Therefore, it satisfies the static equilibrium conditions: the sum of the forces (3.68), and torques (3.69) of the system is zero.

Similar to figure 3.9, an analogy to soft-margin SVM is shown in figure 10.

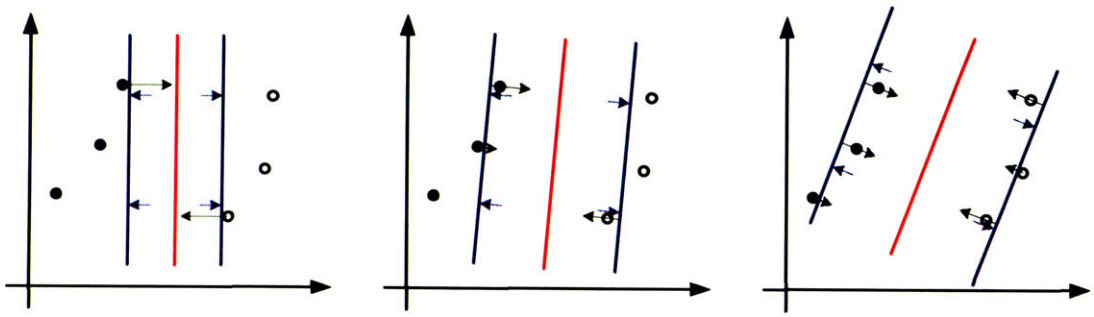


Figure 3.10 Soft-Margin Support Vector Machine

The analogy also applies to inseparable problems because we do not make any assumption on separable while derive (3.68) and (3.69). The inseparable problem is treated by introducing slack variables, and in the mechanics analogy this can be thought of as making the obstacles flexible. Without slack variables, the margin can not move passing any obstacles. When slack variables are used, a margin can pass through obstacles but will be pulled back by those obstacles, as shown in figure 3.11.

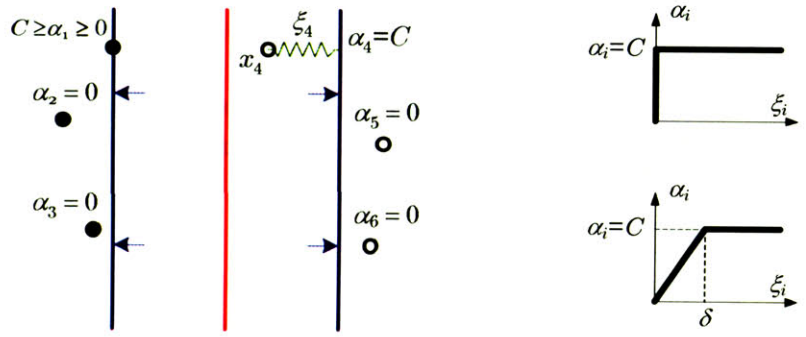


Figure 3.11 Mechanics Analogy of Slack Variables

Recall that if a sample $\underset{\sim}{x}_i$ is not a support vector, the corresponding $\alpha_i = 0$. Also, if $\underset{\sim}{x}_i$ is a support vector, (3.26) shows that if $\underset{\sim}{x}_i$ lays on the boundary, i.e., $\xi_i = 0$, then $\alpha_i = C$, and if $\underset{\sim}{x}_i$ is is not on the boundary, i.e., $\xi_i > 0$, we have $C \geq \alpha_i \geq 0$. These can be summarized as the material property of a spring in the analogy. As shown in the upper-right in figure 3.11, a slack variable defined using (3.8) represents an elastic perfectly-plastic spring that has an elastic region $\rightarrow 0$. A different definition of slack variable will result in a different material.

4. Incorporating Prior Knowledge

This chapter review ways of incorporating prior knowledge into machine learning followed by proposed approach.

4.1 Discriminative Learning in Generative Model

Discriminative learning algorithms such as SVM and boosting have become standard techniques of applied machine learning, because they have achieved record benchmark results in a variety of domains. On the other hand, those "tabula rasa" methods assume the learning algorithm has no prior knowledge except for the representation of hypotheses, and that it must learn solely from training data. It is well understood that prior knowledge can reduce sample complexity and improve learning accuracy. The focus of this work is on discriminative learning, in particular, support vector machine. However, when solving a practical problem, we usually have some prior knowledge about the system and the outcome we expect, such as the function family of the underlying data generating function, preference for a smooth function, the correlations between certain dimensions of the inputs, or the correlation between the outputs. Discriminative learning, based on risk minimization, does not allow most types of prior knowledge to be incorporated into the process of estimation explicitly. On the contrary, generative model based approaches, such as Bayesian learning methods and hidden Markov models, has an intuitive way of applying prior knowledge.

4.1.1 Bayesian learning

Bayesian learning provides a probabilistic approach to inference. It is based on the assumption that the data generating processes are governed by certain probability distributions, and an optimal decision can be made by these probability distributions, the observed data, and our prior knowledge or beliefs about the distributions. Features of Bayesian learning methods includes [6],

1. The estimated probability of a hypothesis can be incrementally decreased or increased by each observed training examples. This is more flexible than algorithms that completely eliminate a hypothesis if it is found to be inconsistent with any single example, or algorithms that regards certain data as outliers and ignores their contributions completely.

2. The final probability of a hypothesis is determined by the combination of observed data together with prior knowledge or beliefs. In Bayesian learning, prior knowledge is provided by asserting a prior probability for each candidate hypotheses, and a prior distribution over observed data for each possible hypothesis.

3. Bayesian methods can accommodate hypotheses that make probabilistic predictions. Therefore, new instances can be classified by combining the predictions of multiple hypotheses and weighted by their probabilities.

4. Even in cases where Bayesian methods prove computationally intractable, they can provide a standard of optimal decision making against which other practical methods can be measured.

Two major difficulties in applying Bayesian based approach. First, the computational cost increases linearly with the number of candidate hypotheses, and is often intractable in the general cases where the hypotheses space is very large. We have seen in chapter 3, on the other hand, SVM is able to search efficiently in an infinite hypotheses space. Second, prior knowledge, in the form of probability distributions, about the data generating process is necessary for Bayesian based approaches. If those probability distributions are not known in advance, they need to be estimated based on background knowledge and previously available data, which may impose significant bias.

4.1.2 Discriminative Model VS. Generative Model

Recall that the goal of machine learning, given a sample data set d, is to find a model h that maximize the probability of seeing the data set, i.e., finding $\arg\max_h p(h|d)$. Both discriminative learning and generative learning share this same goal, but with very different approach. Generative model assumes d is generated by the unknown underlying model h with probability distribution $p(d|h)$, hence the name generative model. As discussed in Chapter 2, $p(h|d)$ is calculated from $p(d|h)$ and prior knowledge $p(h)$ using Baye's rule. In contrast, generative learning does not use the concept of $p(d|h)$, instead, it solves the posterior probability distribution $p(h|d)$ directly by searching through the entire model space.

To make this concrete, let us illustrate the concepts with a classification problem. Assume we are given a data set of 5 points, 3 of them are positive and the other 2 are negative (positive samples are shown as disks and negative points are shown as triangles in figure 4.1) The task is to find a function that separates the positive from the negative. From the generative model's point of view, all samples are generated from a unknown underlying function h. We consider two functions, a vertical line at one-third of the width to the left boundary (Figure 4.1a) and a vertical line at one-third of the width to the left boundary (4.1b).

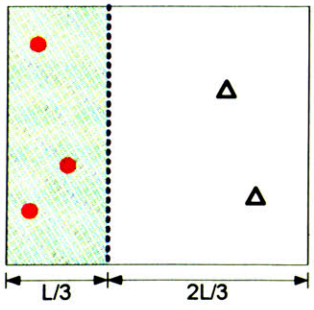
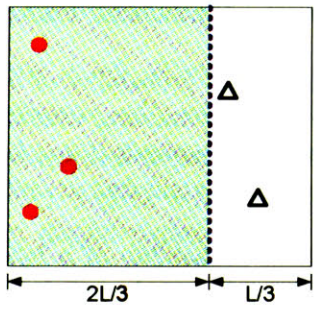


Figure 4.1a classifier closer to the left



4.1b classifier closer to the right

Assume samples are drawn IID, and the vertical line shown in figure 4.1a is the actual underlying discriminant function, the probability of getting three positive and two negative samples from the population is $p(d|h_L) = (1/3)^3(2/3)^2 = 1.65\times10^{-2}$ (the positive area is one-third the total sample space). If figure 4.2a represents the real situation, then the probability $p(d|h_R) = (1/3)^2(2/3)^3 = 3.29\times10^{-2}$. In this case, we prefer h_R over h_L following the maximum likelihood. However, if we have some prior evidence or domain knowledge that the true distribution

is closer to the area enclosed by the RBF function, we can assign prior probability to the two discriminant functions based on the prior belief, say, $p(h_L)=0.6$ and $p(h_R)=0.2$, as depicted in Figure 4.2.

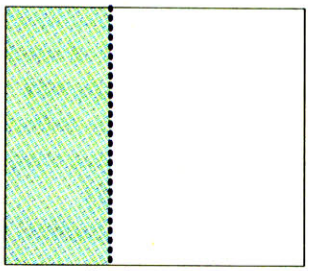
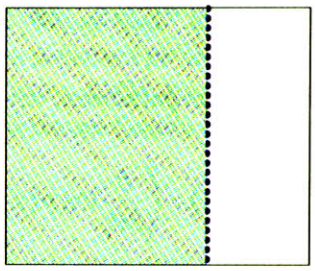


Figure 4.2a Prior Probability of h_L



4.2b Prior probability of hypothesis h_R

Using Baye's rule, we can calculate the posterior probability and chose the maximum a posterior hypothesis. Now the posterior probability of $p(d \mid h_L) = 1.65\times10^{-3}\times0.6 = 9.88\times10^{-3}$ is higher than the posterior probability of $p(d \mid h_R) = 0.81\times10^{-3}\times0.5 = 3.29\times10^{-3}\times0.2 = 6.58\times10^{-3}$, i.e., we prefer h_L over h_R as a MAP classifier.

On the other hand, when a discriminative learning algorithm, such as SVM is used, we do not assign any probability distribution to either the samples or the candidate hypothesis. What we do is we search the best hypothesis from the hypothesis space (in this case all vertical lines), $h \in \mathcal{H}$, based on the regularization criteria of choice (here a maximum margin criteria is used).

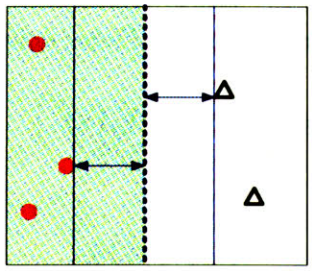


Figure 4.3 Support vector machine classifier

Note that in the generative model, prior knowledge of each hypothesis can be assigned individually and explicitly. Discriminative model also needs prior knowledge. However, it can only be assigned implicitly in a global scale, for instance, the maximum margin criterion used here or a smoothness criterion used by RBF based kernels.

4.1.3 Incorporating Generative Model into Kernel

Since defining a kernel function is equivalent to defining a mapping function from sample space to feature space, it thus defines a metric relation between the training samples implicitly. Jaakkola et al. [17] suggest that these relations should be defined directly from a probability model.

Let us first define the *score*. The score V is the partial derivative with respect to some parameter θ, of the log of the probability distribution of a random variable x.

$$V_\theta(x) = \frac{\partial}{\partial\theta}\log p(x\mid\theta) = \frac{1}{p(x\mid\theta)}\frac{\partial}{\partial\theta}p(x\mid\theta) \tag{4.1}$$

If there is more than one parameter, the score map is the derivative map of $\log p(x\mid\underset{\sim}{\theta})$

$$V_{\underset{\sim}{\theta}}(x) = \left\langle \frac{\partial}{\partial\theta_1}\log p(x\mid\underset{\sim}{\theta}),\cdots,\frac{\partial}{\partial\theta_n}\log p(x\mid\underset{\sim}{\theta}) \right\rangle = \nabla_{\underset{\sim}{\theta}}\log p(x\mid\underset{\sim}{\theta}) \tag{4.2}$$

The *Fisher information* is defined as the variance of the score. Note that the expected value of the score is

$$\begin{aligned} E\left[V_\theta(x)\right] &= \int_x \left(\frac{1}{p(x\mid\theta)}\frac{\partial}{\partial\theta}p(x\mid\theta)\right)p(x\mid\theta)dx \\ &= \frac{\partial}{\partial\theta}\int_x p(x\mid\theta)dx = \frac{\partial}{\partial\theta}1 \\ &= 0 \end{aligned} \tag{4.3}$$

Therefore the variance of the score is

$$E\left[V_\theta(x)^2\right] = E\left[\frac{1}{p(x\mid\theta)}\frac{\partial}{\partial\theta}p(x\mid\theta)\right] \tag{4.4}$$

And the vector version of Fisher information, called *Fisher information matrix*, is

$$J(\underset{\sim}{\theta}) = E\left[V_{\underset{\sim}{\theta}}(x)V_{\underset{\sim}{\theta}}(x)^T\right] \tag{4.5}$$

That is,

$$J_{i,j}(\underset{\sim}{\theta}) = E\left[\frac{\partial}{\partial\theta_i}\log p(x\mid\underset{\sim}{\theta})\cdot\frac{\partial}{\partial\theta_j}\log p(x\mid\underset{\sim}{\theta})\right] \tag{4.6}$$

The Fisher information J is a metric that can be used to measure the amount of information that an observable random variable x carries about the unobservable parameter θ, if x is generated from $p(x\mid\underset{\sim}{\theta})$. A kernel defined using (4.6) is called a Fisher kernel [17], where the Fisher information matrix serves as a measure of the difference in the generative process between each pair of training samples.

4.2 Kernels from Feature Maps

There are two ways of constructing a kernel: by defining a symmetric semi-positive definite matrix, which in turns defines a feature map implicitly; or by defining a feature map function, and which defines a corresponding dot product kernel. It is known as the "kernel trick" that given an algorithm that is formulated in terms of a positive definite kernel, one can construct an alternative algorithm by replacing the kernel with another positive definite kernel. Taking SVM for example, replacing a kernel with another equivalent to solving the same problem in different feature spaces, the question is whether the space makes sense to the problem at hand.

4.2.1 Kernel Definition

A function $\mathcal{K}$ is a kernel if it can be written as an inner product for all $\underset{\sim}{x}_i, \underset{\sim}{x}_j \in \mathbf{X}$,

$$\mathcal{K}\left(\underset{\sim}{x}_i, \underset{\sim}{x}_j\right) = \underset{\sim}{\phi}\left(\underset{\sim}{x}_i\right)^T \underset{\sim}{\phi}\left(\underset{\sim}{x}_j\right) \tag{4.7}$$

where $\underset{\sim}{\phi}$ is a mapping from $\mathbf{X}$ to an feature space $\mathcal{H}$

$$\underset{\sim}{\phi} : \underset{\sim}{x}_i \mapsto \underset{\sim}{\phi}\left(\underset{\sim}{x}_i\right) \in \mathcal{H} \tag{4.8}$$

The second kernel definition can be stated as following: A function $\mathcal{K}$ is a kernel if for any finite set of training examples, $\underset{\sim}{x}_1, \cdots \underset{\sim}{x}_m$, the $m \times m$ kernel matrix $\mathbf{K}$, where $\mathbf{K}_{ij} = \mathcal{K}\left(\underset{\sim}{x}_i, \underset{\sim}{x}_j\right)$, is positive semi-definite. A kernel matrix can be written as

$$\mathbf{K}_{ij} = \mathcal{K}\left(\underset{\sim}{x}_i, \underset{\sim}{x}_j\right) = \underset{\sim}{\phi}\left(\underset{\sim}{x}_i\right)^T \underset{\sim}{\phi}\left(\underset{\sim}{x}_j\right) \tag{4.9}$$

or

$$\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^T \quad \text{where} \quad \mathbf{\Phi}_{m \times d} = \begin{Bmatrix} \underset{\sim}{\phi}\left(\underset{\sim}{x}_1\right)^T \\ \vdots \\ \underset{\sim}{\phi}\left(\underset{\sim}{x}_m\right)^T \end{Bmatrix} = \begin{bmatrix} \phi_1(x_1) & \cdots & \phi_d(x_1) \\ \vdots & \ddots & \vdots \\ \phi_1(x_m) & \cdots & \phi_d(x_m) \end{bmatrix} \tag{4.10}$$

therefore the kernel matrix $\mathbf{K}$ is a Gram matrix, i.e., it contains all possible inner products of the $\underset{\sim}{\phi}\left(\underset{\sim}{x}_i\right)$. Note that we often map the samples from its original sample space to a much higher feature dimensional space $\mathcal{H}$, thus d can be a large number and to calculate (4.9) explicitly needs $\Theta(d^2)$ operations, and (4.10) requires $\Theta(m^2 d^2)$ operations.

4.2.2 Evaluating Explicitly Constructed Kernel

Although every positive semi-definite kernel matrix defines a feature space implicitly, the embedded feature space usually cannot be expressed in explicit form and the meaning is not clear. It is then not easy to incorporate prior knowledge into kernel methods beyond general properties such as

invariants. For many practical engineering and business problem, we often have some understanding about the relation among different types of data. In these cases, constructing kernels explicitly is the best way to capture the domain knowledge. A feature mapping defined in (4.8) can be written as

$$\underset{\sim}{\phi}(\underline{x}_i)=\begin{Bmatrix}\phi_1(\underline{x}_i)\\ \vdots \\ \phi_d(\underline{x}_i)\end{Bmatrix} \tag{4.11}$$

The major drawback of using explicitly constructed kernel is that the evaluation of the kernel matrix $\mathbf{K}$ is inefficient. To address this issue, three approaches are proposed to speed-up the algorithm. First, we randomly sample and quantize entries in the kernel and the evaluation of the randomized proximate kernel is more efficient. Second, we reduce the size of features by a regularization based feature selection approach. Third, the samples are partitioned and trained separately during the training process. The last point will be discussed in detail in the next chapter.

The idea is to pick a subset $\hat{\mathbf{X}} \subset \mathbf{X}$ randomly, and construct a kernel matrix $\hat{\mathbf{K}}$ base on $\hat{\mathbf{X}}$ as an approximation of the exact kernel $\mathbf{K}$ constructed using the full set $\mathbf{X}$.

As an probably approximately correct (PAC) approach, we first define an error ε and a probability δ so that

$$P\left(C\left(\mathbf{K},\hat{\mathbf{K}}\right)\le\varepsilon\right)\ge 1-\delta \tag{4.12}$$

where C can be any criterion or dissimilarity measurement that has an output between 0 to 1 (zero means exactly the same in terms of C). Therefore, any $\varepsilon \le 0.5$ and $\delta \le 0.5$ is an option better than random guess. Assume $\mathbf{K}$ has m entries and $\hat{\mathbf{K}}$ has $\hat{m} \subset m$ entries. Then, how large the subset needs to be so that the maximum entry in $\hat{\mathbf{K}}$ is close enough to the maximum entry in $\mathbf{K}$?

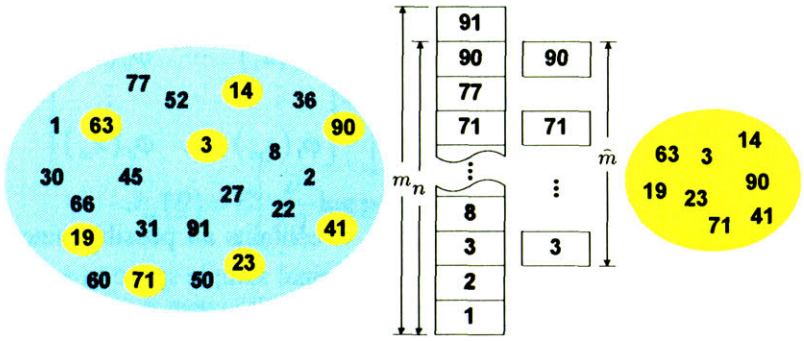


Figure 4.5. Randomly sampled subset

Assume we sort all entries in the full set $\mathbf{K}$, and that all entries in the subset $\hat{\mathbf{K}}$ are selected from the lower n entries in $\mathbf{K}$, as shown in figure 4.5. The probability of this to be true is,

$$\frac{\text{All possible selections from } n}{\text{All possible selections from } m} = \frac{\binom{n}{\hat{m}}}{\binom{m}{\hat{m}}} = \frac{\frac{n!}{(n-\hat{m})!\hat{m}!}}{\frac{m!}{(m-\hat{m})!\hat{m}!}} = \frac{n\cdot(n-1)\cdots(n-\hat{m}+1)}{m\cdot(m-1)\cdots(m-\hat{m}+1)} < \left(\frac{n}{m}\right)^{\hat{m}} \quad (4.13)$$

When this is not true, there must be at least one entry in $\hat{\mathbf{K}}$ that is outside the lower n subset, i.e., the maximum $\hat{k}_{\max} \in \hat{\mathbf{K}}$ is greater than at least n entries in the full set $\mathbf{K}$,

$$P\left(\hat{k}_{\max} > \text{at least } n \text{ entries out of the total } m \text{ entries in } \mathbf{K}\right) \geq 1 - \left(\frac{n}{m}\right)^{\hat{m}} \quad (4.14)$$

Same probability and proof apply if we are looking for the minimum. For instance, we have an array of m = 1,000,000 entries, and we want to find the maximum in the array. How many samples do we need to get a probably approximately result that has 95% chance to be greater than 95% of all entries? Given P = 0.95 and n / m = 0.95 in (4.14), we can calculate the sample size $\hat{m} \geq 59$. Notice that there is no assumption in the underlying distribution and the result is independent of the sample size.

Now assume the criterion is to check the closeness of the averages of the two sets, instead of the closeness between the maximums. For this, we use the well-known Hoeffding bound [18] that bounds the deviation of the sample mean from its expected value,

$$P\left(\left|E(\mathbf{X}) - \frac{1}{m}\sum_{i=1}^{m} x_i\right| \geq \varepsilon\right) \leq 2\exp\left(-\frac{2m^2\varepsilon^2}{\sum_{i=1}^{m}(b_i - a_i)^2}\right) \quad (4.15)$$

where $\hat{m}$ is the sample size, and each random variable $x_i \in \mathbf{X}$ is bounded in $a_i \leq x_i \leq b_i$.

To use Hoeffding bound in our case, we need a slight change so that it bounds the deviation is between full set and subset rather than the deviation between full set and the whole population (the expected value). Instead of bounding $E(\mathbf{X}) - \frac{1}{m}\sum_{i=1}^{m} x_i$, now we are bounding $\frac{1}{m}\sum_{i=1}^{m} x_i - \frac{1}{\hat{m}}\sum_{i=1}^{\hat{m}} x_i$.

Since $\hat{\mathbf{X}} \subset \mathbf{X}$, we can rewrite $\frac{1}{m}\sum_{i=1}^{m} x_i - \frac{1}{\hat{m}}\sum_{i=1}^{\hat{m}} x_i$ as,

$$\begin{aligned}
\frac{1}{m}\sum_{i=1}^{m} x_i - \frac{1}{\hat{m}}\sum_{i=1}^{\hat{m}} x_i &= \sum_{i=1}^{m}\frac{x_i}{m} - \sum_{i=1}^{\hat{m}}\frac{x_i}{\hat{m}} = \sum_{i=1}^{\hat{m}} x_i\left(\frac{1}{m} - \frac{1}{\hat{m}}\right) + \sum_{i=\hat{m}+1}^{m}\frac{x_i}{m} \\
&= \frac{1}{m}\left[\sum_{i=1}^{\hat{m}}\left(1 - \frac{m}{\hat{m}}\right)x_i + \sum_{i=\hat{m}+1}^{m} x_i\right] \\
&= \frac{1}{m}\sum_{i=1}^{m}\left[\left(\left(1 - \frac{m}{\hat{m}}\right)H(\hat{m} - i) + H(i - \hat{m} - 1)\right)x_i\right]
\end{aligned} \quad (4.16)$$

where H is the unit step function with

$$H(n)=\begin{cases}1, & n\geq 0\\ 0, & n<0\end{cases} \tag{4.17}$$

Let the result of (4.16) to be the new random variable $y_i \in \mathbf{Y}$, i.e.,

$$y_i=\left[\left(1-\frac{m}{\hat{m}}\right)H(\hat{m}-i)+H(i-\hat{m}-1)\right]x_i \tag{4.18}$$

And each y_i is bounded by $a_i' \leq y_i \leq b_i'$ where

$$\begin{aligned} a_i' &= \left[\left(1-\frac{m}{\hat{m}}\right)H(\hat{m}-i)+H(i-\hat{m}-1)\right]a_i \\ b_i' &= \left[\left(1-\frac{m}{\hat{m}}\right)H(\hat{m}-i)+H(i-\hat{m}-1)\right]b_i \end{aligned} \tag{4.19}$$

and if all x_i is bounded in the same range $a \leq x_i \leq b$

$$\sum_{i=1}^{m}(b_i'-a_i')^2=\hat{m}\left(1-\frac{m}{\hat{m}}\right)^2(b-a)^2+(m-\hat{m})(b-a)^2=m\left(\frac{m}{\hat{m}}-1\right)(b-a)^2 \tag{4.20}$$

Note that

$$\begin{aligned} E(\mathbf{Y}) &= \frac{1}{m}\sum_{i=1}^{m}E(y_i)=E\left(\frac{1}{m}\sum_{i=1}^{m}y_i\right)=E\left(\frac{1}{m}\sum_{i=1}^{m}x_i-\frac{1}{\hat{m}}\sum_{i=1}^{\hat{m}}x_i\right) \\ &= \frac{1}{m}\sum_{i=1}^{m}E(x_i)-\frac{1}{\hat{m}}\sum_{i=1}^{\hat{m}}E(x_i)=\frac{m}{m}E(x_i)-\frac{\hat{m}}{\hat{m}}E(x_i)=0 \end{aligned} \tag{4.21}$$

Finally, by substituting (4.18)~(4.21) into (4.15) we have,

$$P\left(\left|\frac{1}{m}\sum_{i=1}^{m}x_i-\frac{1}{\hat{m}}\sum_{i=1}^{\hat{m}}x_i\right|\geq\varepsilon\right)\leq 2\exp\left(-\frac{2m^2\varepsilon^2}{m\left(\frac{m}{\hat{m}}-1\right)(b-a)^2}\right)=2\exp\left(-\frac{2\varepsilon^2}{(b-a)^2}m\frac{\frac{\hat{m}}{m}}{1-\frac{\hat{m}}{m}}\right) \tag{4.22}$$

Given a specific probability $P\leq\delta$ and the difference ε, we can calculate the sample size required to achieve this accuracy is

$$\frac{\hat{m}}{m}=\frac{\gamma}{1+\gamma} \quad \text{where} \quad \gamma=\frac{(b-a)^2\ln(2/\delta)}{2m\varepsilon^2} \tag{4.23}$$

when m is large, γ is small and the ratio $\hat{m}/m$ is small. To give a numeric example, assume the full set has 1,000,000 samples and each sample is normalized within [0, 1]. Let $\delta = 0.05$ and $\varepsilon = 0.025$, the subset needs 2943 samples, less than 0.3% of the full size. As shown in figure 4.6, the number of samples required in the subset $\hat{m}$ is almost independent of the number of total samples in the full set m when m is large. The bound is not at all useful when m is small, and in fact we do not need an approximation when m is small.

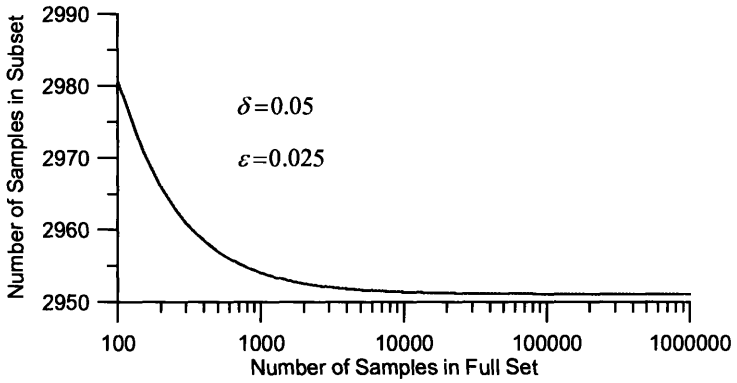


Figure 4.6. Number of samples required in the subset

So far we have been working on the samples $\mathbf{X}$, and now we will move on to the kernel $\mathbf{K}$. Recall that each entry in the kernel matrix $\mathbf{K}$ is given by (4.9). When calculate explicitly, it is very computational expensive when samples are transformed into a high-order feature space. To speed-up the calculation, we randomly pick $\hat{d}$ features out of the total d features, and rescale the sum,

$$\mathcal{K}(\underset{\sim}{x}_i, \underset{\sim}{x}_j) = \sum_{l=1}^{d} \phi_l(\underset{\sim}{x}_i)\phi_l(\underset{\sim}{x}_j) \approx \frac{d}{\hat{d}} \sum_{l=1}^{\hat{d}} \phi_l(\underset{\sim}{x}_i)\phi_l(\underset{\sim}{x}_j) = \hat{\mathcal{K}}(\underset{\sim}{x}_i, \underset{\sim}{x}_j) \tag{4.24}$$

Since the features are selected randomly, we have,

$$E\left(\hat{\mathcal{K}}(\underset{\sim}{x}_i, \underset{\sim}{x}_j)\right) = \mathcal{K}(\underset{\sim}{x}_i, \underset{\sim}{x}_j) \tag{4.25}$$

Again, we can use (4.15) to bound the deviation between the actual kernel entry and the approximation.

$$\begin{aligned} P\left(\left|\mathcal{K}(\underset{\sim}{x}_i, \underset{\sim}{x}_j) - \hat{\mathcal{K}}(\underset{\sim}{x}_i, \underset{\sim}{x}_j)\right| \geq \varepsilon\right) &= P\left(\left|E\left(\hat{\mathcal{K}}(\underset{\sim}{x}_i, \underset{\sim}{x}_j)\right) - \frac{1}{\hat{d}} \sum_{l=1}^{\hat{d}} d\phi_l(\underset{\sim}{x}_i)\phi_l(\underset{\sim}{x}_j)\right| \geq \varepsilon\right) \\ &\leq 2\exp\left(-\frac{2\hat{d}^2\varepsilon^2}{\sum_{i=1}^{\hat{d}}(b_i - a_i)^2}\right) \end{aligned} \tag{4.26}$$

If the value of all features are bounded within [a, b], the difference between an entry in the true kernel and the proximate kernel is less than ε with probability at least $1 - \lambda$ where,

$$\lambda = 2\exp\left(-\frac{2\hat{d}\varepsilon^2}{(b-a)^2}\right) \tag{4.27}$$

If we requires that with probability at least $1-\delta$ that all $m \times m$ entries in the true kernel and proximate kernel have differences within ε, then,

$$\delta \geq \frac{(1+m)m}{2} \cdot 2\exp\left(-\frac{2\hat{d}\varepsilon^2}{(b-a)^2}\right) \tag{4.28}$$

because $1-(1-\lambda)^n \leq n\lambda$ (cf. (2.12)) and a kernel matrix is symmetric. Hence, the number of features we need, $\hat{d}$,

$$\hat{d} \geq \frac{(b-a)^2}{2\varepsilon^2}\left(\ln\left((1+m)m\right) - \ln\delta\right) \tag{4.29}$$

Given $b-a=1$, $\varepsilon=0.05$, $m=100{,}000$, and $\delta=0.05$, we need $\hat{d}=5{,}205$. The relation between $\hat{d}$ and m is shown in figure 4.7.

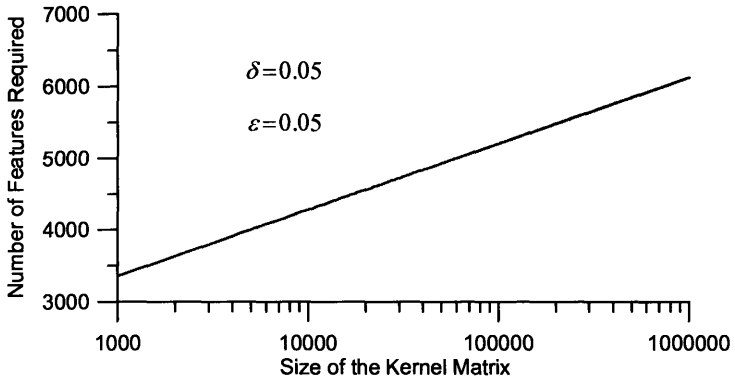


Figure 4.7. Number of features required

Recall that if each sample $\underset{\sim}{x}_i$ has 100 components, an entry in a linear kernel can be evaluated in 100 operations. A polynomial kernel of degree 5 can be evaluated in almost the same operations, using implicit kernel mapping. If evaluated explicitly, it requires $\binom{105}{5} = 96{,}560{,}646$ operations. With proposed method 5,205 operations are needed.

4.2.3 Feature Selection with Kernel

A prediction suffix tree (PST) is not an efficient choice when the number of features is large. To consider all features jointly, we can use the idea of regularization to calculate the weights and select the corresponding basis functions, as discussed in lecture. We try to extend that idea into SVM,

since it is more efficient and consistent to the SVM concept. By looking at the result of the primal form of SVM,

$$f(\underset{\sim}{x}_i) = \underset{\sim}{w}_1^T \underset{\sim}{\phi}(\underset{\sim}{x}_i) + w_0 \tag{4.30}$$

each component w_{1j} in $\underset{\sim}{w}_1$ can be thought of as the weight of the corresponding basis function $\phi_j(\underset{\sim}{x}_i)$. Since the goal is to reduce the number of features in the sample space, it is nature to use linear kernel in this feature selection scenario. With a linear kernel, the discriminant function is simply,

$$f(\underset{\sim}{x}_i) = \underset{\sim}{w}_1^T \underset{\sim}{x}_i + w_0 \tag{4.31}$$

and the number of components in $\underset{\sim}{w}_1$ is equal to the number of features. The primal representation of SVM problem is to minimize equation (3.9), which is also what we want in feature selection. We know that the regularization penalty depends on its derivative at $w_i \doteq 0$, therefore the regularization term in SVM only forces the value of each component $|w_i|$ to be small, but does not set it to zero. However, to be able to take advantage of the dual representation of SVM, we cannot change the regularization term from minimizing $\|\underset{\sim}{w}_1\|^2$ to $\|\underset{\sim}{w}_1\|$. The simplest way to get around this is to set a threshold, and remove those basis functions associate with weights smaller than the threshold. For linear kernel, the basis functions are the features themselves

In deriving the dual form of SVM we know,

$$\frac{\partial}{\partial \xi_i} J(\underset{\sim}{w}, \xi, \alpha, \beta) = \underset{\sim}{w}_1 - \sum_{i=1}^{\ell} \alpha_i y_i \underset{\sim}{x}_i = 0 \tag{4.32}$$

therefore we can calculate $\underset{\sim}{w}_1$ easily using the solution of the dual form. In fact, many of the α_i are zeros (the non-support vectors) and we need only to calculate,

$$\underset{\sim}{w}_1 = \sum_{i=1}^{n_{SV}} \alpha_i y_i \underset{\sim}{x}_i \tag{4.33}$$

4.2.4 The Curse of Dimensionality and the Blessing of Large Number

The curse of dimensionality imposes severe restriction on many learning algorithms. The original SVM gets around the problem by using implicit kernels that do not need to calculate dot products explicitly, thus avoid the curse. The proposed approach needs to perform the dot products explicitly but because we only seek for a probably approximately correct result, we can obtain a good estimate of the true kernel with high probability using only fraction of the total number of the features. This is due to the large number of the samples and features. Therefore, the proposed approach is cursed and blessed at the same time. One may argue that it is better not to be cursed at all, but the difference is that now domain knowledge can be incorporated directly and that can greatly improve the overall learning results.

5. Design of the Learning and Simulation Framework

I have introduced the general SVM algorithms and its in chapter 3, and I have also discussed the functions of kernels in chapter 4. What is lacking is a computational framework that puts it all together. The challenges of implementing such a framework are two fold. On the *theoretical* side, the framework should be able to solve large scale problems in reasonable time and an efficient solver is required. On the *architectural* side, the framework should be lightweight, flexible, and can be easily integrated into existing systems. We achieve the first goal by introducing an efficient approximation solver that takes advantage of the complementary slackness property in support vector machine. The second goal is fulfilled by implementing as an assembly using a platform-independent language that also supports modern network protocols.

In this chapter, I first review some existing solvers used in SVM. Then, I introduce the algorithm I proposed and used in the implementation. Next, I discuss the architecture and implementation of the proposed framework in detail. Finally, a nontrivial example is used to demonstrated the framework and as a verification of concept.

5.1 Solving the Quadratic Programming Problem

As discussed in chapter 3, to solve the support vector machine optimization problem efficiently, we first transform the primal form (3.9) to its dual form (3.21), where only equality and constant constraints exists. However, the dual form still can not be solved analytically except for very small training data sets. (or the training data is separable and the support vectors are known beforehand [19]). In most real world cases, the problem must be solved numerically.

Mathematically, training SVM is equivalent to solving a quadratic programming (QP) problem with linear constraints (3.21), and the number of variables is equal to the number of training data points. Solving QP problem has been studied for a long time, and there are many general purpose algorithms available. However, the optimization problem becomes challenging when the number of variables exceeds several thousands, and old existing solvers can not handle the size of a practical machine learning problem. Take interior point method, one of the most widely used general QP algorithms, as an example, current implementations are mostly based on Mehrotra's predictor-corrector technique [20], where a Cholesky decomposition of the normal equation is used to perform the Newton iteration, together with some heuristics to estimate the penalty parameter. The bottleneck of this algorithm is in the Cholesky decomposition, which requires n^2 memory space to store the matrix and $\Theta(n^3)$ to decompose the matrix. It is clear that when n exceeds several thousands this approach will become infeasible.

Many QP problems can be speed up by taking advantages of using efficient algorithms of sparse Cholesky decomposition. Unfortunately, the QP problem generated by training a SVM almost always results in a dense matrix. Therefore, special algorithms are required when dealing large scale SVM. Strategies proposed in previous work to solve the large scale QP problem take advantages of the fact that the number of support vectors, n_{SV}, is often much smaller than the number of training samples. Therefore, by identifying those non-support vectors heuristically, the problem size can be

reduced to proportional to the number of support vectors, not the total training samples. Those algorithms still have two major drawbacks. First, not all problems have a small n_{SV}/n ratio. A "difficult" problem (problem with high generalization error) will always have a high n_{SV}/n ratio, since it is the upper bond of the generalization error (LOO, leave one out). Second, even for a simpler problem, when the total training samples is really large (say, > 100,000), those algorithms are still doomed to fail.

For small and moderate size problem, interior point method is often the method of choice because it has higher accuracy. However, it is not suitable for large-scale problems. Current standard approaches of improving training efficiency for large datasets is based on chunking and different versions of working set techniques.

The basic idea behind Chunking algorithm, which is introduced in [21], is based on the observation that the solution of the embedded QP problem remains the same if the rows and columns of the matrix that correspond to zero Lagrange multiplies are removed. The problem is that we do not know which vectors will be support vectors before solving the problem. Therefore the approach is performed in a heuristic way that starts with an arbitrary subset. The small set is used to train the SVM, and discard the non-support vector data after it is trained. Unused data from the full set is then added back iteratively, and only support vectors are kept after each step. This approach reduces the size of the QP problem from n^2 to n_{SV}^2.

Chunking method fails when the number of support vectors is still too large to fit into the memory. Working set [22] solves this problem by breaking a large QP problem into sets of smaller QP problems. Only a subset of the QP problem is optimized while keeping the other variables constant. This decomposition algorithm uses a constant size matrix for the sub-problem therefore the size does not exceed a certain limit. However a numerical QP solver such as interior point method is still required.

The idea behind Sequential Minimal Optimization (SMO) is similar to that of working set. The difference is, in SMO, the idea is taken to the extreme where only two samples are in the set and the QP problem can be solved analytically, and no QP solver is needed. SMO can handle very large training sets less than $\Theta(n^2)$ time with $\Theta(n)$ memory, while chunking and the original working set require $\Theta(n^3)$ time.

5.2 Speeding-up with Large Training Sets

5.2.1 Re-sampling in Sample Space

As shown in section 4.3.2, a PAC result can be obtained using a subset with moderate size when the resample is performed randomly. A set contains 10,000 dots and 10,000 cross is used as an example, and the goal is to separate the two classes using SVM with Gaussian kernel. Note that the data shown in the figures has been reduced to 1/10 the actual data size in order to help visualization, i.e., only 1,000 samples in each class. Two subsets are selected randomly, each with 10,000 samples.

The full set and each subset are trained separately, and the results are shown in figure 5.1 and figure 5.2. The support vectors are enclosed by squares.

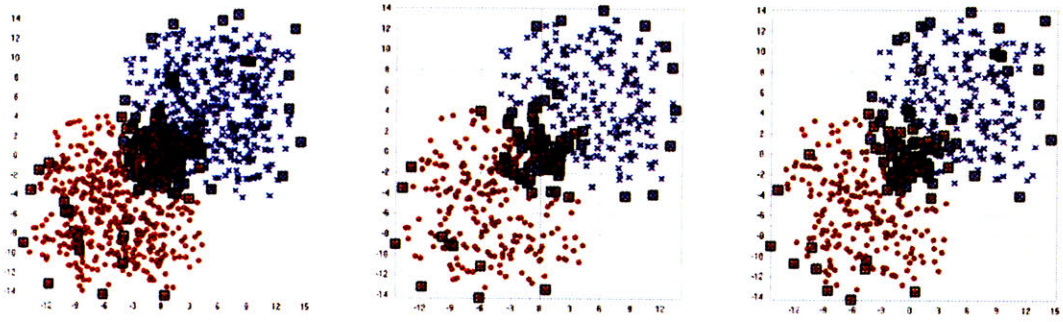


Figure 5.1 Training Samples. Full Set (Left), Partition 1 (Middle), and Partition 2 (Right)

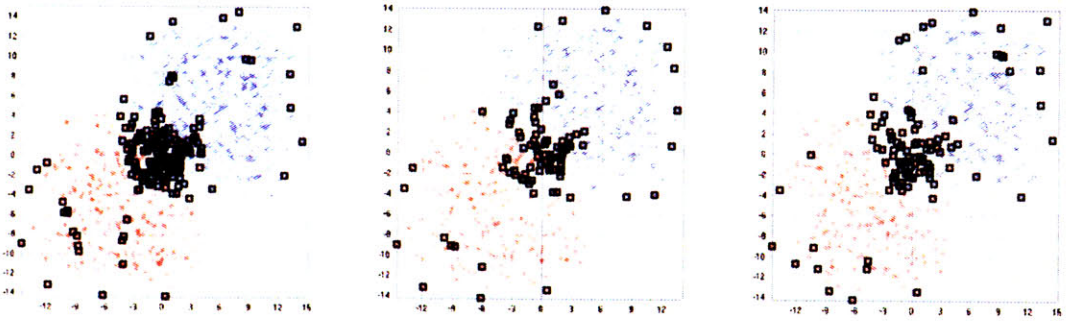


Figure 5.2 Support vectors. Full set, 173 (Left), Subset 1, 88 (Middle), and Subset 2. 103 (Right)

The training time required for the full set is 25.6 seconds, while the time required for each subset is 8.2 seconds and 8.7 seconds, respectively.

5.2.2 Combining Weak Learners

A straightforward way of combining the results is to train a final model with the support vectors from each subset. Because the number of support vectors in each subset is relatively small, this final step can be performed efficiently. The result and comparison is shown in figure 5.3.

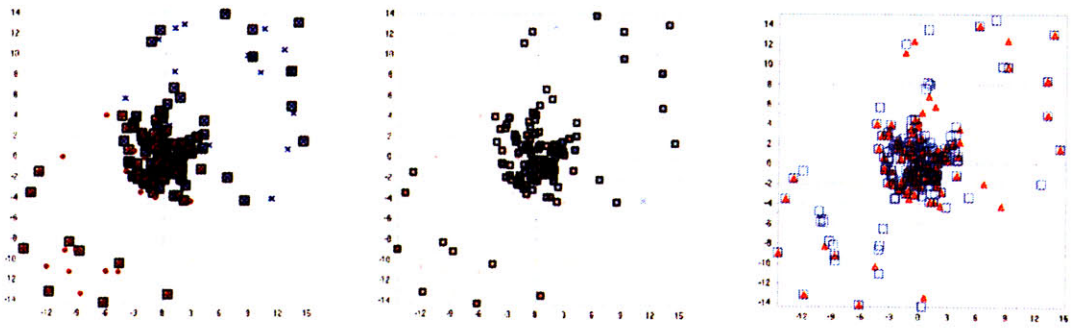


Figure 5.3. Combining the support vectors (#SV=152)
Left / Middle: Combined result
Right: Comparison between SVs from the full set (square)
and combined result (triangle)

Support vector machine is a batch learning algorithm, i.e., all training data must be trained in one pass. If there are new data available, the model needs to be retrained with the whole data set to take the new information into account. Combining support vectors from the subsets becomes inefficient when SVM is used in an adaptive learning scheme.

Alternatively, we can collect the outputs of each subset and combine them. The simplest way to combine all models is to take the average,

$$\mathbf{M}^s(\underset{\sim}{x}_i) = \frac{1}{s}\sum_{j=1}^{s} M_j(\underset{\sim}{x}_i) \tag{5.1}$$

where $\mathbf{M}^s$ is the combined model with s members, each member M_j is a SVM model trained using the j-th subset. Considering each model may have different quality, it makes sense to use a weighted average,

$$\mathbf{M}^s(\underset{\sim}{x}_i) = \sum_{j=1}^{s} \alpha_j M_j(\underset{\sim}{x}_i) \tag{5.2}$$

Where $\alpha_j \geq 0$, and $\sum_{j=1}^{s} \alpha_j = 0$. Equation (5.2) can be viewed as a voting mechanism that each component has a weight $\alpha_j \geq 0$. Equation (5.1) is a special case of (5.2) where all models carry the same weight.

Now the problem is how to determine α_j. First we define an exponential loss,

$$Loss\left(y_i, M_j(\underset{\sim}{x}_i)\right) = \exp\left(-y_i M_j(\underset{\sim}{x}_i)\right) \tag{5.3}$$

For a classification problem $y_i = \pm 1$ and $M_j(\underset{\sim}{x}_i) \in [-1,1]$. If y_i agrees with $M_j(\underset{\sim}{x}_i)$, the $-y_i M_j(\underset{\sim}{x}_i)$ is negative and the loss is small; while if the two disagree, $-y_i M_j(\underset{\sim}{x}_i)$ is positive and the loss is higher. The loss function we just defined is bounded in $Loss\left(y_i, M_j(\underset{\sim}{x}_i)\right) \in \left[e^{-1}, e^{+1}\right]$.

Suppose now we have added $s-1$ SVM models in the voting system already, i.e.,

$$\mathbf{M}^{s-1}(\underset{\sim}{x}_i) = \sum_{j=1}^{s-1} \alpha_j M_j(\underset{\sim}{x}_i) \tag{5.4}$$

When adding one more model to the system, our goal is to find an α_s so that the updated system M^s

$$\mathbf{M}^{s}(\underset{\sim}{x}_i) = \mathbf{M}^{s-1}(\underset{\sim}{x}_i) + \alpha_s M_s(\underset{\sim}{x}_i) \tag{5.5}$$

has a smaller loss than the original system $\mathbf{M}^{s-1}$. For each training sample the exponential loss is,

$$\begin{aligned} Loss\left(y_i, \mathbf{M}^{s}(\underset{\sim}{x}_i)\right) &= \exp\left(-y_i \mathbf{M}^{s}(\underset{\sim}{x}_i)\right) \\ &= \exp\left(-y_i\left(\mathbf{M}^{s-1}(\underset{\sim}{x}_i) + \alpha_s M_s(\underset{\sim}{x}_i)\right)\right) \\ &= \exp\left(-y_i \mathbf{M}^{s-1}(\underset{\sim}{x}_i)\right) \cdot \exp\left(\left(-y_i \alpha_s M_s(\underset{\sim}{x}_i)\right)\right) \\ &= Loss\left(y_i, \mathbf{M}^{s-1}(\underset{\sim}{x}_i)\right) \cdot \exp\left(\left(-y_i \alpha_s M_s(\underset{\sim}{x}_i)\right)\right) \end{aligned} \tag{5.6}$$

Note that at step s, $Loss\left(y_i, \mathbf{M}^{s-1}(\underset{\sim}{x}_i)\right) \equiv L_{s-1}(\underset{\sim}{x}_i)$ is a constant

The total loss for all training samples is,

$$J(\alpha_j) = \sum_{i=1}^{m} Loss\left(y_i, \mathbf{M}^{s}(\underset{\sim}{x}_i)\right) = \sum_{i=1}^{m}\left(L_{s-1}(\underset{\sim}{x}_i) \cdot \exp\left(\left(-y_i \alpha_s M_s(\underset{\sim}{x}_i)\right)\right)\right) \tag{5.7}$$

To minimize (5.7) we take the derivative with respect to α_s,

$$\frac{\partial}{\partial \alpha_s} J(\alpha_s) = -\sum_{i=1}^{m}\left(L_{s-1}(\underset{\sim}{x}_i) \cdot y_i M_s(\underset{\sim}{x}_i) \cdot \exp\left(\left(-y_i \alpha_s M_s(\underset{\sim}{x}_i)\right)\right)\right) = 0 \tag{5.8}$$

Note that since $y_i M_s(\underset{\sim}{x}_i) = 1$ if y_i agrees with $M_s(\underset{\sim}{x}_i)$ and $y_i M_s(\underset{\sim}{x}_i) = -1$ if the two disagree, we can separate (5.8) into

$$-\sum_{i,\, y_i M_s(\underset{\sim}{x}_i)=+1}\left(L_{s-1}(\underset{\sim}{x}_i) \cdot 1 \cdot \exp(-\alpha_s)\right) - \sum_{i,\, y_i M_s(\underset{\sim}{x}_i)=-1}\left(L_{s-1}(\underset{\sim}{x}_i) \cdot -1 \cdot \exp(\alpha_s)\right) = 0 \tag{5.9}$$

therefore,

$$\alpha_s = \frac{1}{2}\ln\left(\frac{\sum_{i,\, y_i M_s(\underset{\sim}{x}_i)=+1} L_{s-1}(\underset{\sim}{x}_i)}{\sum_{i,\, y_i M_s(\underset{\sim}{x}_i)=-1} L_{s-1}(\underset{\sim}{x}_i)}\right) \tag{5.10}$$

The meaning of $\sum_{i,\, y_i M_s(\underset{\sim}{x}_i)=+1} L_{s-1}(\underset{\sim}{x}_i)$ is the number of the training samples classified correctly by the new model M_s weighted by L_{s-1}, the loss from the current combined model $\mathbf{M}^{s-1}$. Similarly, $\sum_{i,\, y_i M_s(\underset{\sim}{x}_i)=-1} L_{s-1}(\underset{\sim}{x}_i)$ is the number of the training samples classified incorrectly by the new model M_s

weighted by L_{s-1}. Recall that α_s is the weight of the vote of the new model M_s. When more training samples are classified correctly by M_s, the numerator is larger and the denominator is smaller in (5.10) and the value of α_s is larger, i.e., M_s has a stronger vote. Also, if a training sample $\underset{\sim}{x}_i$ is already correctly classified by $\mathbf{M}^{s-1}$, the loss L_{s-1} is smaller and that particular training sample will have less effect on the value of α_s. Therefore, α_s has a larger value when M_s can correctly classify training samples that are misclassified by $\mathbf{M}^{s-1}$.

To add a new model, we simply use (5.10) to calculate the weight of its vote and append the new model to the current model. To evaluate (5.10) is a process of testing, with both $\mathbf{M}^{s-1}$ and M_s. Testing is an $\Theta(n)$ operation and can be performed quickly.

Before the new model is taken into account it can be thought of as has a zero weight, i.e., $\alpha_j = 0$. We want to know how much we can reduce the overall loss by considering the contribution from the new model, therefore we need,

$$\frac{\partial}{\partial \alpha_j} L(\alpha_j)\Big|_{\alpha_j=0} = \sum_{i=1}^{m}\left(\exp\left(-y_i M_{s-1}(\underset{\sim}{x}_i)\right)\cdot\left(-y_i M_j(\underset{\sim}{x}_i)\right)\right) \tag{5.11}$$

Because $\alpha_j \geq 0$, the derivative needs to be negative so the overall loss will decrease.

5.3 Implementation

5.3.1 .NET

The SVM implementation is written in Microsoft C# and it is compiled as a .NET class library that can be used by other .NET applications easily. To test the library and visualize the result (figure 5.4), a few simple 2D graph functions are also provided in the implementation. All examples shown in this work are built on top of this SVM library.

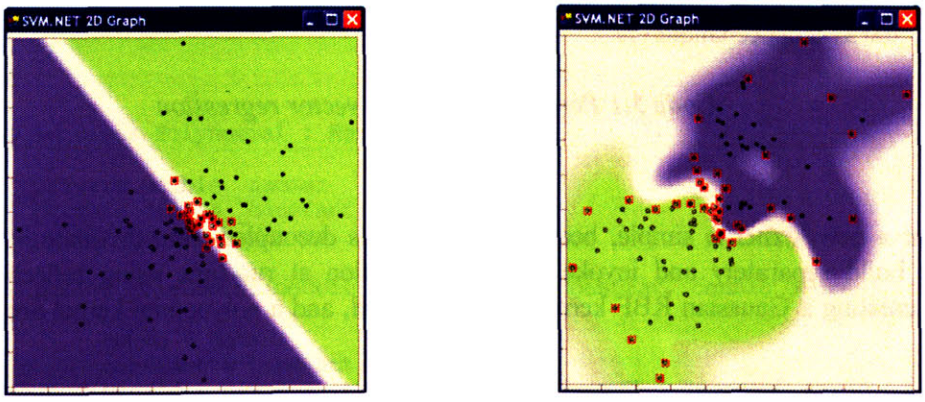


Figure 5.4. Classifying two inseparable classes in 2D
(Left: with linear kernel. Right: with RBF kernel. The squares highlight the support vectors)

The assembly can also be embedded into database, such as Microsoft SQL Server 2005 Analysis Services, through constructing a managed plug-in algorithm.

5.3.2 Using SVM.NET

The use of the library is straightforward. In the current version, a singleton SVM object is used to build a SVM model from the given training data. Code examples are given below,

```
private void DoRegression()
{
    double[] y = new double[100];
    double[][] x = new double[y.Length][];

    Random rand = new Random();
    for (int i = 0; i < y.Length; i++)
    {
        x[i] = new double[] { 2 * Math.PI / y.Length * i };
        y[i] = Math.Sin(x[i][0]) + rand.NextDouble() / 2;
    }

    SVMachine SVM = SVMachine.Instance;
    SVM.SvmType = SVMType.EPSILON_SVR;
    // SVM.Parameter.C = 100;
    SVM.Kernel = new Gaussian(1);

    SVM.buildSVMModel(y, x);

    _data = new DataTable();
    DataColumn X1 = new DataColumn("X1", typeof(double));
    DataColumn Y1 = new DataColumn("Y1", typeof(double));
    DataColumn Y2 = new DataColumn("Y2", typeof(double));

    _data.Columns.AddRange(new DataColumn[] { X1, Y1, Y2 });

    DataRow dr;
    for (int i = 0; i < y.Length; i++)
    {
        dr = _data.NewRow();
        dr[X1] = x[i][0];
        dr[Y1] = y[i];
        dr[Y2] = SVM.predict(x[i]);
        _data.Rows.Add(dr);
    }
    _data.AcceptChanges();
}
```

Code 5.1 Performing support vector regression

To implement a new kernel is simple, because the kernel is decoupled from the main application, thus can be build separately and invoke by the application at run-time using reflection. Code examples of creating a Gaussian RBF kernel, a linear kernel, and a polynomial kernel are shown in below.

```
public class Gaussian : KernelFunction
{
    private double _sigma;
    private double _variance;
```

```
    public Gaussian(double sigma)
    {
        _sigma = sigma;
        _variance = Math.Pow(sigma, 2);
    }

    public double Sigma
    {
        get { return _sigma; }
        set
        {
            _sigma = value;
            _variance = Math.Pow(value, 2);
        }
    }

    public double Variance
    {
        get { return _variance; }
    }

    public override double evaluate(sparseVec x1, sparseVec x2)
    {
        return Math.Exp(-L2_norm_of_X1_minus_X2(x1, x2) / (2 * _variance));
    }
}
```

Code 5.2 Gaussian Kernel

```
public class Linear : KernelFunction
{
    public Linear()
    {
    }

    public override double evaluate(sparseVec x1, sparseVec x2)
    {
        return dot(x1, x2);
    }
}
```

Code 5.3 Linear Kernel

```
public class Polynomial : KernelFunction
{
    private double _gamma;
    private double _constant;
    private double _degree;

    public Polynomial(double gamma, double constant, double degree)
    {
        _gamma = gamma;
        _constant = constant;
        _degree = degree;
    }

    public override double evaluate(sparseVec x1, sparseVec x2)
    {
```

```
            return Math.Pow(_gamma * dot(x1, x2) + _constant, _degree);
        }
    }
```

Code 5.4 Polynomial Kernel

The lightweight assembly can also be used as web-service or as the intelligent core of a larger service. An sample web page using the web-service is shown in figure 5.5, where the interface is implemented using Ajax (Asynchronous JavaScript and XML) to produce a richer web user interface.

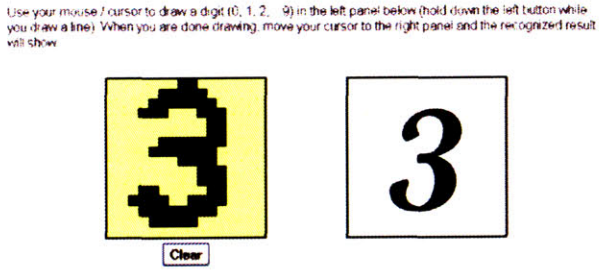


Figure 5.5 Handwriting Recognition in a Web Page

5.4 Examples

5.4.1 Generalization Error Estimation

Before we examine the proposed framework with examples, we need to introduce some performance metrics. Several methods are frequently used in estimating generalization errors in a given learning methods, namely, leave-one-out (LOO) cross validation, general cross validation (leave-n-out, or LNO), and bootstrap. We use the term cross-validation loosely here for all generalization error estimation methods based on re-sampling, but we need to point out that no method is suitable for all type of learning problems and the study of cross-validation itself is quite subtle and has a long history [23, 24]. Cross-validation is often used to estimate the generalization error of a given model, or used for model selection by choosing one of several models that has the smallest estimated generalization error. In SVM, it is used to test the generalization error as well as select parameters best suits a given problem heuristically.

Suppose we are given a learning algorithm L and a set of m training samples, $\{z_1,\cdots,z_m\} = \{(x_1,y_1),\cdots,(x_m,y_m)\}$. To estimate the generalization error of the algorithm L, we can train the algorithm with $\{z_1,\cdots,z_{m-1}\}$ and use the trained model to predict the unseen sample z_m. To get a better estimation, we can simply repeat the process m times, and each time we leave a different sample out and use the rest of the set as training samples. The estimated generalization error given by LOO cross-validation is,

$$P(error) = \frac{\text{Number of times where } L(x_i) \neq y_i}{m} \tag{5.12}$$

It is intuitive to understand that LOO cross-validation often works better for estimating generalization error for continuous error functions than discontinuous error functions such as classification problems. For classification problems, an n-fold cross-validation, or LNO, is often preferred. LNO is done by a similar fashion as LOO, the only difference is in LNO, the training sample set is first randomly divided into n subsets, each with sample size equal $m_{s1},\cdots m_{sn}$, and $\sum_{i=1}^{n} m_{si} = m$. Each subset usually has same or similar size. Then, we leave a subset out as testing samples and use the rest as training samples. The cross-validation error given by LNO is also given by (5.12). Note that while both LOO and LNO have the same formula, the reported error is usually different. A value of 10 for n is often used, because a very small n leads to a cross-validation training set much smaller than the full training set and may result in unwanted bias. If n=m, LNO becomes LOO. Last, bootstrapping in its simplest form is very similar to LNO, except instead of using subsets (which can be thought of as randomly picking samples *without* replacement), bootstrapping pick its testing sets by re-sampling every time *with* replacement from the full training sample set.

5.4.2 Handwriting Recognition

MNIST is a standard pattern recognition benchmark sample at AT&T. It contains a training set of 60,000 examples, and a test set of 10,000 examples. Each character is represented as a 16 pixels by 16 pixels, 256-level gray scale image. A small part of the training set is shown graphically in figure 5.6.

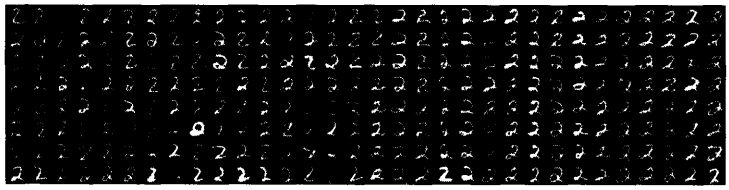


Figure 5.6 MNIST Handwriting Digit Set

The classification test error using neural networks ranges from 1.6% (2 layers with 800 hidden units) to 12% (1 layer with no hidden unit), the current record low error is about 0.4%. A SVM with Gaussian RBF kernel can achieve a test error equal to 1.4%. Considering invariant in small rotation, translation, and/or size, SVM has a test error around 0.56%. When using proposed speed-up algorithm with 10 partitions, it results in a 1% higher error rate than normal SVM but is 10 times faster. If the data is fed incrementally, since the proposed method can perform incremental learning, which is more than 2 orders faster than retrain with all data.

5.4.3 Email Screening

A training corpus consists of 1,379 emails, among them 747 are spams, is used in my experiments. The feature set of each email is extracted using bag-of-words and the standard TF-IDF score. We first scan through all emails in the training set and record every distinct word and its document frequency (DF, the number of different emails it appears) in a table of a relational database management system (RDBMS). Then, for each email, we list all its distinct words and their term frequency (TF, the number of times it appears in that email) in another table. By joining the two tables (in standard SQL language, *table_DF* LEFT OUTER JOIN *table_TF*) we can calculate the

final TF-IDF score. The definition of IDF has a few different versions, and we use IDF = log(1+N/DF), where N is the total number of emails (N=1,379). We ignore any word less than 2 characters long, and the training set yielded 22,025 distinct words. That is, every email has a feature with 22,025 components, the value of each component is determined by its TF-IDF score, and most of them are zeros. Other information about the emails is also recorded in the database and can be used as features. For example, the size of the email, the number of attachments, email address of the sender, the number of people in the mailing list... etc. Those could be useful in improving the accuracy of the spam filter, but they are not studied in this example.

A 5-fold cross validation is used to test the accuracy of the approach. In each testing, the training examples are randomly divided into 5 groups with the same size, and cross validation is performed and the final result is obtained by average the testing results. With this initial feature set, the experiment shown that SVM can achieve as high as 96% accuracy in distinguishing between spam and non-spam. In next section, I first switch gear to feature selection.

5.4.4 Feature Selection Using SVM

As mentioned, the initial features of each email are determined by the traditional TFIDF method. It surely gives a very high accuracy but we are also sure there are a lot of redundant features in the 22,025 components long feature set. In this section, I introduce a possible way to refine the features set.

Although in the dual form of SVM we are facing a quadratic programming problem whose computational complexity is proximately proportional to the square of the number of training samples ℓ, not the number of features, reducing the number of features is nevertheless helping to improve the performance. For example, when evaluating a kernel function, such as a linear kernel or a Gaussian kernel, we need to perform a lot of dot products between feature vectors, and this step is also time consuming if the number of features is very large (in fact in SVM solver we often cache the results to increase the efficiency)

Besides, the storage of long features is also a problem. Considering that a spam screening company that has 10,000 users, each has 2,000 emails, and each email has 20,000 features, and each feature (a real number) needs 4 bytes to store, the total adds up to as large as 1,600 GB (most of the features are zeros so the actually size could be an order or two smaller). It is not terribly large, but it is large enough to slow down the performance of an industrial database. Last but not least, there are likely to be many noises in the features. If we can identify those noises and eliminate them from the feature set, we can not only improve the computational efficiency but also increase the accuracy.

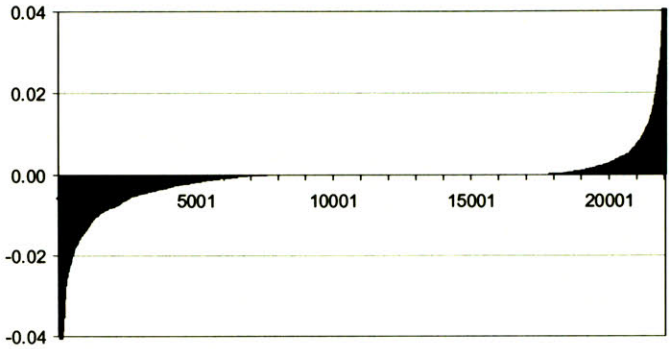


Figure 5.7 Sorted weights of a typical $\underset{\sim}{w}_1$ vector
The sign came from the class label (+1, -1), thus the weight should be defined as the absolute value.

To confirm the feature selection scheme, we add two additional features to the original 22,025 components feature set: 1.) the class label itself (extremely informative), and 2.) a constant (non-informative feature) at the end. No surprise, the "label feature" received a dominantly large weight (0.666) while the sum of all weights is 0.703, and the "non-informative feature" was assigned a zero weight. The result shows that using linear kernel in feature selection is promising. More experiments and discussing will be given in the next section.

To see the effect of using different kernels in the classification approach, we studied the two most widely used kernels: linear kernel and radial basis kernel (RBF).

	Linear	Radial Basis Function		
		σ^2=100	σ^2=1000	σ^2=10000
1 (265)	254	221	253	255
2 (266)	257	216	248	253
3 (266)	242	209	242	259
4 (266)	247	226	253	250
5 (266)	243	218	250	261
Average	93.5%	82.0%	93.8%	96.2%

Table 5.1. Kernel comparison using 22,025 features

We can easily see from table 5.1 that linear kernel is as good as RBF kernels. Further, RBF kernel has an additional parameter σ^2 that need to be selected carefully and cannot be determined a prior (we can, however, give a ballpark figure by looking at the number of features, their values, and the sparseness).

Considering that we have a very high dimension and a very sparse feature space, it is reasonable to assume that the vectors (emails) are easy to be separated, and that a hyperplane is sufficient for this job. To put it another way, in the example, a linear hyperplane has VC dimension equal to 22,025 + 1 while we have only 1,379 training documents, the complexity of linear kernel is more than enough. Therefore, a more complex kernel like RBF does not have much reason to be more accurate.

Using the feature selection scheme described in chapter 4, we reduced the number of features to roughly 1/10 each time, and test the accuracy of using the reduced feature set with linear kernel. The results are shown in table 5.2.

	Number of Features			
	22,025	**2,163**	**233**	**33**
1 (265)	254	251	238	228
2 (266)	257	247	245	240
3 (266)	242	255	242	234
4 (266)	247	256	236	222
5 (266)	243	244	240	234
Average	93.5%	94.3%	90.7%	87.1%

Table 5.2. Accuracy versus number of features

We can see that with only 33 words, the filter still retains a very high accuracy in the 5-fold cross validation. However, there is a change that we have over-fitted the data. Because in the feature selection process, we keep the first 10% features (words) associates with the highest weights. After repeat the process three times, only features that classify the training set well are retained. In other words, the 33 words long feature set may be too specific to the training set, and this situation can not be revealed by cross validation, since the way we do cross validation only helps in sample wise, not in feature wise.

Another thing worth noticing is that if we cut the features directly from 22,025 to 33, we will get much worse results, as shown in table 5.3,

	Progressive	Direct
1 (265)	228	155
2 (266)	240	168
3 (266)	234	164
4 (266)	222	162
5 (266)	234	159
Average	87.1%	60.1%

Table 5.3. Progressive reduction versus directly cut

This is simply because if we cut the features directly, many of the emails do not have any of the words in the short feature set, both spam and non-spam. Those zero features is obviously useless, thus result in the poor accuracy. This also supports the previous comment that the progressive reduction may have over-fitted the feature set. After we reduce the features to 33, we try RBF kernel with the new feature set and the results are listed in table 5.4.

	Linear	Radial Basis Function		
		σ^2=10	σ^2=100	σ^2=1000
1 (265)	228	240	234	191
2 (266)	240	233	229	206
3 (266)	234	238	244	191
4 (266)	222	237	232	205
5 (266)	234	244	225	189
Average	87.1%	89.7%	87.6%	73.9%

Table 5.4. Kernel comparison using 33 features

We can see that the reduced feature set also works well with RBF kernel, as they are attributes of the documents, not attributes of kernel. Therefore, linear kernel servers as a good way of feature selection, and does not restrict us to using only linear kernel in classification. Finally, figure 5.8, 5.9, and 5.10 shows the components of the $\underset{\sim}{w}_1$ vector after we add 2 additional features (feature 34 and feature 35) into the 33 components feature set and perform the linear SVM classification.

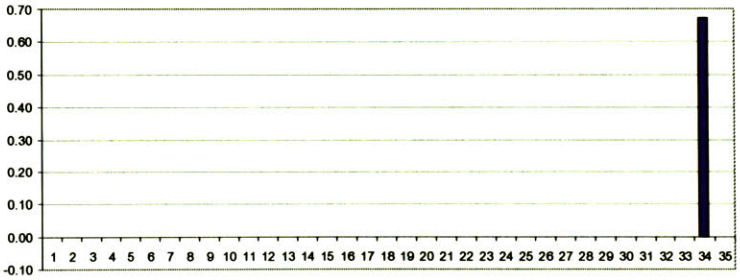


Figure 5.8 34: correct label. 35: constant (non-informative)

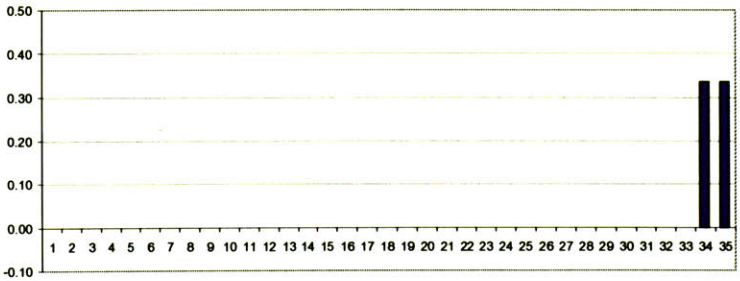


Figure 5.9 34: correct label. 35: correct label.

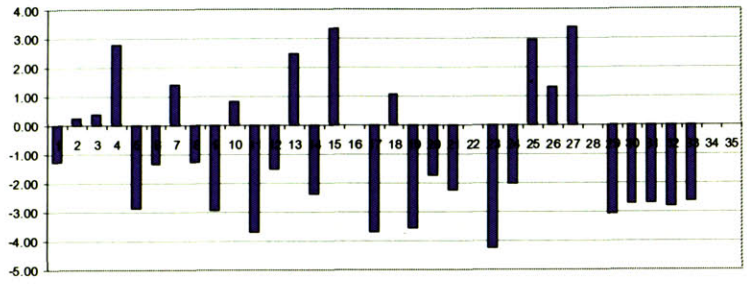


Figure 5.10 34: constant. 35: constant.

It is clear that the linear kernel has assigned suitable weights to the more informative features, as we would expect. It is worth noticing that when we have an extremely informative feature, the overall values of components become much smaller (compare figure 5.8 to figure 5.10). This is due to the fact that SVM can separate the two classes easily with that feature, thus a much larger margin can be obtained, i.e., a much smaller $\|\underset{\sim}{w}_1\|^2$.

6. Structural Health Monitoring

The process of implementing a damage detection strategy is referred to as structural health monitoring (SHM), and can be categorized into five stages [25]: (1) detecting the existence of damage, (2) locating the damage, (3) identifying the type of the damage, (4) determining the severity of the damage, (5) predicting the remaining service life of the structure. Nondestructive testing methods such as ultrasonic scanning, acoustic emission, radiography inspection, and magnetic field methods have been widely applied in localized damage detection, but have the limitation that the occurrence of damage needs to be known a prior and the damaged area is accessible for inspection. To overcome these difficulties and to evaluate the global state of the potentially damaged structure, vibration-based damage detection methods have been proposed and many researches have been conducted in this field during the past decade. Literature reviews can be found in [26, 27]. The basic reasoning behind all vibration-based damage detection is that the stiffness, mass, or energy dissipation behavior of the structure changes significantly when damage occurs. These properties can be detected and measured by monitoring the dynamic response of the system using various sensors and sensor networks [28].

Traditional vibration-based damage identification applications rely on detailed finite element models of the undamaged structures. Damage diagnosis is made by comparing the modal responses, such as frequencies and mode shapes, measured from the potentially damaged structure and calculated from the model of the undamaged structure. This system identification approach has been shown to be very accurate provided the model can produce robust and reliable modal estimates, and large amount of high quality data is available [29]. These two requirements cannot always be met in the field, and pattern recognition based approaches have been proposed [30-33]. Instead of building models from physical properties of the structures, pattern recognition based methods construct statistical models from the vibration response data directly. This reduces the complexity of the modeling process, in the cost of losing physical meaning of the model. Also, these methods have been shown to be accurate in damage detection and are less sensitive to data quality; however, several problems still remain. First, how to choose a statistical model is unclear. For example, methods that use autoregressive models (AR/ARX) may not be able to fit the vibration data well because the model gives only linear approximations to the response behavior. On the other hand, complex statistical models using artificial neural network (ANN) [34] are less efficient, and have little control over the generalization bound and result in overfitting [6], i.e., they may fit the history data perfectly but have no guarantee on the future data. Second, learning based methods such as ANN require intensive computation resources, and do not scale well to large scale problems. Recently, support vector machine (SVM), has also been applied to perform supervised, binary classifications in damage detection [32, 35]. Training SVM is mathematically equivalent to solving a convex quadratic programming (QP) problem, which does not have local minima and can be trained faster than algorithm that does. Third, a statistical model is often constructed through a supervised learning process, which requires response data from structure in various damage patterns to be known in a prior, and is not feasible in most practical situations. Fourth, models constructed solely based on the observed data does not take advantages of domain knowledge we already have about the structure, and results in a non-interpretable mathematical model sensitive to data quality.

In this chapter, single-class SVM and support vector regression (SVR) are used to perform unsupervised damage detection and eliminate the need of knowing the response from damaged structure a prior. Also, structural dynamics is incorporated into the otherwise discriminative learning algorithm during feature selection. The use of this prior knowledge leads to a more robust model that combines the advantage of both analytical and inductive learning.

6.1 Damage Detection Using SVM Based Approaches

SVM has been applied in various pattern recognition fields and it is not new to introduce SVM into SHM. Two extensions of SVM, single-class SVM and SVR, are used in this chapter. Both algorithms are able to perform unsupervised pattern analysis tasks; therefore only dynamic responses of undamaged structures are require for constructing the statistical models when applied to damage detection. Also, structural dynamics is incorporated into the learning scheme through explicitly constructed kernel.

SVM is originally a supervised, batch learning algorithm, and has been applied in the SHM field [32, 35] performing supervised classification tasks. A major challenge is that data corresponding to different damage patterns of the structure is often not available in practical situations, thus unsupervised learning methods are more desirable [36]. One way to transform SVM into unsupervised learning is, instead of finding a hyperplane that maximize the margin between two classes in the RKHS, we maximize the distance from a hyperplane to the origin, this is often referred to as single-class SVM [37]. The corresponding optimization problem becomes,

$$\begin{aligned} &\text{minimize: } \frac{1}{2}\left\|\underset{\sim}{\mathbf{w}}_1\right\|^2 + \frac{1}{\nu\ell}\sum_i^{\ell}\xi_i - \rho \\ &\text{subject to: } \underset{\sim}{\phi}^{\mathrm{T}}(\underset{\sim}{\mathbf{x}}_i)\underset{\sim}{\mathbf{w}}_1 \geq \rho - \xi_i \quad \text{and} \quad \xi_i \geq 0 \end{aligned} \tag{6.1}$$

where $\nu \in (0,1]$ is a parameter analogous to the C introduced in (3.9), and ρ is a offset that will be calculated automatically during the optimization. If a translation invariant kernel is used (e.g. RBF kernel), the goal of single-class SVM can also be thought of as to find small spheres that contain most of the training samples.

The basic idea of SVM, single-class SVM and SVR are summarized in Table 6.1 and Figure 6.1. For illustration, the discriminant function of SVM and single-class SVM are drawn as linear functions. As mentioned, when nonlinear kernels are used, the functions are by no means linear in the input space.

	Maximize	Penalty
SVM	distance between two hyperplanes	misclassified samples
Single-Class SVM	distance between the hyperplane and origin	misclassified samples
SVR	smoothness of the function	samples outside the ε - tube

Table 6.1. Comparison of SVM, single-class SVM and SVR

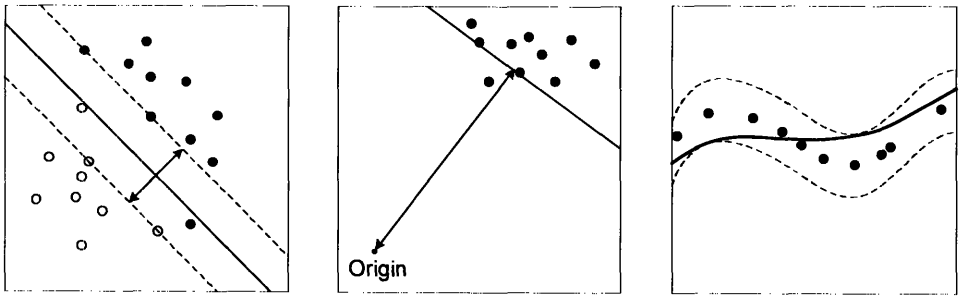


Figure 6.1. Geometric interpretation of SVM, single-class SVM and SVR in 2D

6.2 Damage Detection Using Proposed Approaches

Damage detection using SVR is a 2-step approach. First, a statistical model of the undamaged structure is constructed through learning. Vibration responses corresponding to any external excitation can be used as training examples, as long as the magnitude of the excitation is moderate and the structure remains undamaged. The concept of this step is summarized in figure 2.

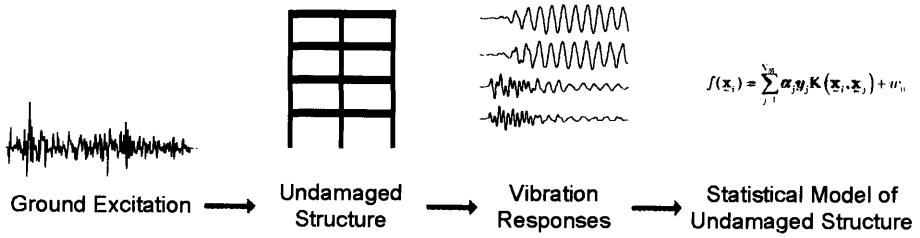


Figure 6.2. Constructing a Statistical Model for Undamaged Structure

Next, when the structure is potentially damaged, e.g., after struck by an earthquake, the model can be used to simulate the vibration responses the structure would have if it is not damaged. Damage detection is conducted by examining the similarity and dissimilarity between responses collected from the potentially damaged structure and responses generated from the model, as illustrated in figure 3.

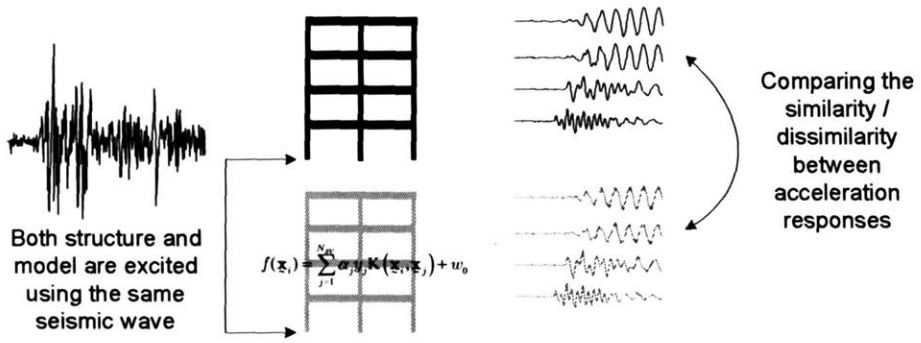


Figure 6.3. Damage Detection Using the Statistical Model

Applying single-class SVM is similar to applying SVR. The only difference is that single-class SVM does not generate theoretical responses in the second step. Instead, it compares the responses from the potentially damaged structure to the original training responses directly. Because the two response sets are generated by different external excitations, the responses need to be normalized and the model needs to take external forces into account. Note that being a novelty detection algorithm, single-class SVM separate one structure status from all others via a discriminant function, and is suitable for detecting the occurrence rather than locating the position of damages in a structure. On the other hand, SVR produces a mathematical model of the undamaged structure, and is capable of indicating the location of damages. Detailed analysis procedure will be given in the numerical studies section.

6.3 Domain Knowledge and Feature Selection

Vibration-based damage detection approaches are grounded on the assumption that the dynamic response of the system will change significantly when damage occurs. It is then essential to have a proper representation of the dynamic response when pattern recognition based algorithms are used for damage detection. I propose using feature selection to achieve such representation and combining domain knowledge with SVM. Note that we use the term feature and pattern interchangeably, because we are performing feature selection in the input / pattern space.

A time series is usually modeled by splitting it into series of windows, and the value at each time point is determined by a set of its previous values. For instance, a series of acceleration response recorded by an accelerometer can be modeled as,

$$a_t = f\left(a_{t-\tau}, a_{t-2\tau}, \ldots, a_{t-m\tau}\right) \tag{6.2}$$

where a_t is the acceleration response at time t, and m and τ are referred to as the embedding dimension and delay time [38], respectively. Through this representation, an acceleration response series can be transformed into a set of label-pattern pairs and used by pattern recognition based methods directly, for instance, Let et al. (2003) and Sohn and Farrar (2001) adopted this representation in damage detection, with an additional residual error term. Note that by using (11), we assume that the acceleration is a function of the previous accelerations at that specific location alone, and does not consider the structure of the asset or the pattern of the external excitation.

Similar to all discriminative learning methods, SVM is a data-driven algorithm that seeks a general model that fits the observed data, and no domain knowledge is required. This is an advantage when applied to less known systems, for instance, detecting damages in an irregularly shaped historical building. Nevertheless, we often have basic understanding about the underlying system and therefore an ideal algorithm for damage detection should incorporate domain knowledge with inductive learning so it can accommodate an unknown level of error in the data. For instance, the incremental dynamic equilibrium equation can be used when selecting features from the response data to take structure dynamics into account,

$$\mathbf{M}\Delta\ddot{\underset{\sim}{u}} + \mathbf{C}\Delta\dot{\underset{\sim}{u}} + \mathbf{K}\Delta\underset{\sim}{u} = \Delta\underset{\sim}{p} \tag{6.3}$$

where $\Delta\underset{\sim}{u}$ is the incremental displacement vector, $\Delta\underset{\sim}{p}$ is the incremental external force vector, and $\mathbf{M}$, $\mathbf{C}$, $\mathbf{K}$ are the mass, damping, and stiffness matrices, respectively. Note that (6.3) is only used to guide the learning, the exact value of $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{K}$ do not need to be known beforehand. Take a 3-story shear building for example, if we ignore the damping and the source of external force is the ground excitation, (6.3) can be written as,

$$\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \begin{Bmatrix} \Delta\ddot{u}_1 \\ \Delta\ddot{u}_2 \\ \Delta\ddot{u}_3 \end{Bmatrix} + \begin{bmatrix} k_1+k_2 & -k_2 & 0 \\ -k_2 & k_2+k_3 & -k_3 \\ 0 & -k_3 & k_3 \end{bmatrix} \begin{Bmatrix} \Delta u_1 \\ \Delta u_2 \\ \Delta u_3 \end{Bmatrix} = -\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix} \Delta\ddot{y}_0 \tag{6.4}$$

where $\Delta\ddot{y}_0$ is the incremental ground acceleration. And if we assume during the small increment time step Δt, the change of acceleration is linear, i.e.,

$$\Delta\dot{\underset{\sim}{u}} = \ddot{\underset{\sim}{u}}\Delta t + \frac{1}{2}\Delta\ddot{\underset{\sim}{u}}\Delta t \tag{6.5}$$

$$\Delta\underset{\sim}{u} = \dot{\underset{\sim}{u}}\Delta t + \frac{1}{2}\ddot{\underset{\sim}{u}}\Delta t^2 + \frac{1}{6}\Delta\ddot{\underset{\sim}{u}}\Delta t^2 \tag{6.6}$$

Equation (6.4) can be rewritten as,

$$\mathbf{M}\left(\frac{6}{\Delta t^2}\Delta\underset{\sim}{u} - \frac{6}{\Delta t}\dot{\underset{\sim}{u}} - 3\ddot{\underset{\sim}{u}}\right) + \mathbf{K}\Delta\underset{\sim}{u} = -\mathbf{M}\Delta\ddot{y}_0 \tag{6.7}$$

and

$$\Delta\underset{\sim}{u} = \left(\mathbf{M}\frac{6}{\Delta t^2} + \mathbf{K}\right)^{-1} \cdot \mathbf{M} \cdot \left(\frac{6}{\Delta t}\dot{\underset{\sim}{u}} + 3\ddot{\underset{\sim}{u}} - \Delta\ddot{y}_0\right) \tag{6.8}$$

Let

$$-\left(\frac{6}{\Delta t^2}\mathbf{M}+\mathbf{K}\right)^{-1}\cdot\mathbf{M}=\mathbf{A}=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{21} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{6.9}$$

Substitute (6.9) into (6.8) we have,

$$\begin{aligned}
\Delta u_1 &= a_{11}\left(\frac{6}{\Delta t}\dot{u}_1+3\ddot{u}_1-\Delta\ddot{y}_0\right)+a_{12}\left(\frac{6}{\Delta t}\dot{u}_2+3\ddot{u}_2-\Delta\ddot{y}_0\right)+a_{13}\left(\frac{6}{\Delta t}\dot{u}_3+3\ddot{u}_3-\Delta\ddot{y}_0\right) \\
\Delta u_2 &= a_{21}\left(\frac{6}{\Delta t}\dot{u}_1+3\ddot{u}_1-\Delta\ddot{y}_0\right)+a_{22}\left(\frac{6}{\Delta t}\dot{u}_2+3\ddot{u}_2-\Delta\ddot{y}_0\right)+a_{23}\left(\frac{6}{\Delta t}\dot{u}_3+3\ddot{u}_3-\Delta\ddot{y}_0\right) \\
\Delta u_3 &= a_{31}\left(\frac{6}{\Delta t}\dot{u}_1+3\ddot{u}_1-\Delta\ddot{y}_0\right)+a_{32}\left(\frac{6}{\Delta t}\dot{u}_2+3\ddot{u}_2-\Delta\ddot{y}_0\right)+a_{33}\left(\frac{6}{\Delta t}\dot{u}_3+3\ddot{u}_3-\Delta\ddot{y}_0\right)
\end{aligned} \tag{6.10}$$

Finally, $\Delta\dot{\underset{\sim}{u}}$ and $\Delta\ddot{\underset{\sim}{u}}$ can be obtained by substituting (6.10) into (6.5), (6.6), and (6.7). Note that $\Delta\ddot{\underset{\sim}{u}}$ is a vector and thus the acceleration at a location is no longer independent from acceleration at other locations. The correlation among accelerations can be learnt from the measurements as we train the statistical model or it can be modeled explicitly based on the layout of the structure to provide more prior knowledge to the learning algorithm. It is obvious a better choice of feature set is

$$\left(\text{label, feature}\right)=\left(\ddot{u}_1^{(t+\Delta t)},\left(\dot{u}_1^{(t)},\dot{u}_2^{(t)},\dot{u}_3^{(t)},\ddot{u}_1^{(t)},\ddot{u}_2^{(t)},\ddot{u}_3^{(t)},\ddot{y}_0^{(t)},\ddot{y}_0^{(t+\Delta t)}\right)\right) \tag{6.11}$$

Note that we are not to predict the structure status at time $t+\Delta t$ given the status at time t, instead, the objective is to construct a model that has similar response as the real structure. Hence, the ground excitation data at time $t+\Delta t$ can also be used as a feature.

Also note that (6.10) is under the assumption that the acceleration response varies linearly during Δt. We can release this constraint by mapping (6.11) to a higher dimensional space either implicitly by using a kernel or explicitly by using a feature mapping function.

6.4 Numerical Studies

In this section, I demonstrate the proposed approaches using a 2-story shear building and the ASCE benchmark problem [39]. In all examples, acceleration responses are first normalized using,

$$\overline{a}_i=\left(a_i-\mu_a\right)/\sigma_a \tag{6.12}$$

where μ_a and σ_a are the sample mean and sample standard deviation of the acceleration responses, respectively. The values of SVM related parameters, such as C, ν, ε are selected based on common practice in pattern recognition and are specified in each example. A Gaussian RBF kernel is used and σ is set to $\sqrt{m}/2$, where m is the number of features, in all examples unless otherwise noticed. Choosing parameters for SVMs is an active research topic, e.g., [40, 41] and it is often done

heuristically in practice. In general, similar results can be obtained in the following examples as long as those parameters are within a reasonable range.

6.4.1 Proof-of-Concept Example: A Two-Story Shear Building

We start with the simple 2-story shear building shown in Figure 4. Damage is modeled by reducing the stiffness of a column by 50%. The 1940 El Centro earthquake is used as the horizontal ground disturbance. Vibration data are collected through accelerometers attached under each floor. The vibration data are recorded at location A and B with 50Hz sampling rate, and the minimum value of τ in (6.2) is therefore 0.02 second. If we choose $\tau = 0.02$ and $m = 100$, the feature vector associates with each label corresponding to a 2-second window in the acceleration response.

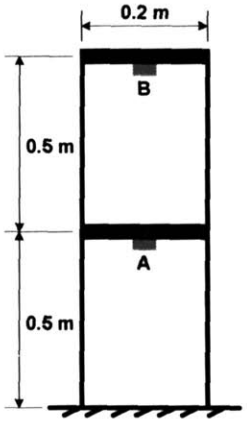


*Figure 6.4. Plane Steel Frame under Traverse Seismic Load (*EI=6.379 Nm^2 *for all columns)*

Three different SVM based approaches are used for damage detection, namely, (1) supervised SVM, (2) single-class SVM, and (3) support vector regression. The proposed damage detection application is implemented in C#, and the solver algorithm for SVM follows [42].

6.4.2 Damage Detection Using Supervised Support Vector Machine

Supervised SVM is only suitable for systems whose damage patterns are known and responses can be measured in advance. In this example, 800 training examples are selected in each structure status (undamaged, 1st floor damaged, and 2nd floor damaged) from the vibration response recorded by the two accelerometers. The label in each example represents the structure status, and the corresponding patterns are extracted using (6.13) with $\tau = 0.02$. The training examples are then fed into SVM (C=100) and the results of a 5-fold cross validation are shown in Table 2. Accuracy is defined as the number of examples classified correctly divided by the number of total testing examples. We can see that SVM is able to detect the occurrence as well as the location of the damage with high accuracy, provided the number of features, i.e., embedding dimension, is long enough.

# of Patterns	CV 1	CV 2	CV 3	CV 4	CV 5	Accuracy
50	97/120	89 / 120	90 / 120	87 / 120	78 / 120	76%
75	111 / 120	111 / 120	119 / 120	117 / 120	116 / 120	96%
100	120 / 120	120 / 120	120 / 120	120 / 120	120 / 120	100%

Table 2. Cross Validation Results (3 structure status; 480 training examples, 120 testing examples)

6.4.3 Damage Detection Using Single-Class Support Vector Machine

Although supervised SVM classification is accurate and easy to implement, in practice we usually do not have vibration response from damaged structure beforehand. In this example, I apply single-class SVM on the same structure using the same features ($m = 50$ for each acceleration response) and kernel. The trade-off parameter ν in (8) is set to 0.1. A single-class SVM model is trained using a set of response data measured from undamaged structure under 5 different seismic loads. This is due to the fact that both external force and structure status can affect the acceleration response, and a model built on one particular load history cannot be generalized well to monitor arbitrary load. To reduce this unwanted effect, we train SVM models using a larger database that consists of mixture of acceleration responses measured from undamaged structure under different seismic loads. By grouping these responses together, we implicitly tell SVM to ignore the differences caused by external force variability.

Response data measured from both damaged and undamaged structure under three specific seismic loads is then used for testing. The seismic loads used in testing are different from those used in the training stage, in order to test how well the model can generalized to unseen data. An outlier detected by single-class SVM means the model considers the response is generated from a different structure status, i.e., a damaged structure. The results are shown below.

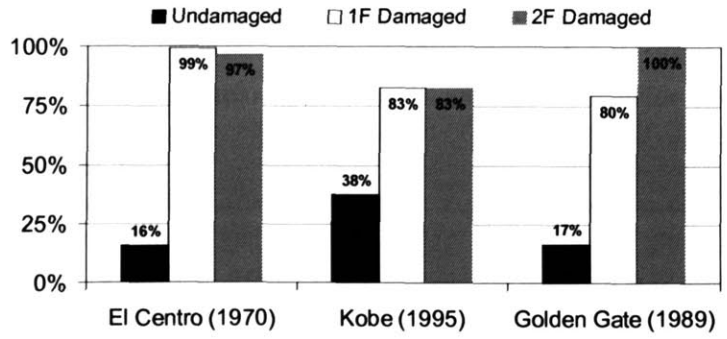


Figure 6.5. Percentage of outliers detected

Figure 6.5 shows that the percentage of outliers increases significantly when damage occurred in the structure, regardless the external force. Damage detection can be done by setting an experimental threshold. Nonetheless, the model detects a significant change when damage occurred in either floor and lacks the ability to tell the location of the damage.

6.4.4 Damage Detection Using Support Vector Regression

The 2-story steel frame shown in Figure 6.2 is used in this example. As in the single-class example, the SVR model is trained using a set of acceleration responses collected from undamaged structure under 5 different seismic loads. Two more seismic loads, El Centro and Golden Gate, are used to generate testing data for each structure status. A polynomial kernel of degree 3 is used in the SVR. The SVR parameters C and ε are set to 100 and 0.1, respectively. Denoting the training error at time t as $\varepsilon_{trn}(t)$, and the testing error at time t as $\varepsilon_{TBD}(t)$. The ratio of the standard deviation of the two errors is defined as the damage-sensitive feature [30],

$$h = \sigma(\varepsilon_{TBD})/\sigma(\varepsilon_{trn}) \tag{6.14}$$

and a experimental threshold limit can be used to indicate the occurrence of damage. The damage detection results are shown in Table 6.3.

Seismic load	El Centro		Golden Gate	
Location of Damage	1F	2F	1F	2F
h (location A, 1F)	**2.984**	1.474	**2.240**	1.173
h (location B, 2F)	1.207	**2.554**	1.244	**2.344**

Table 6.3. Damage Detection in a 2-story Frame using SVR

As expected, the SVR model built from undamaged structure yields significantly higher prediction errors when used to predict the response from damaged structure. When a suitable threshold limit is chosen for the damage-sensitive feature h, the proposed approach is able to indicate both the existence and the location of the damage.

6.4.5 ASCE Benchmark Problem

Structural health monitoring studies often apply different methods to different structures, which make side-by-side comparison of those methods difficult. To coordinate the studies, the ASCE Task Group on Health Monitoring built a 4-story 2-bay by 2-bay steel frame benchmark structure and provided two finite element based models, a 12DOF shear-building and a more realistic 120DOF 3D model [39]. The benchmark problem is studied in this section.

Five damage patterns are defined in the benchmark study, and we apply the SVR detection procedure to the first two patterns: (1) all braces in the first story are removed, and (2) all braces in both the first story and the third story are removed. Acceleration responses of these two damage patterns are generated by using the 12DOF analytical model under ambient wind load. The results of damage detection and localization using damage-sensitive feature h is shown in Table 6.4. The training data is a mixture of acceleration responses obtained from the undamaged structure under 10 different ambient loads. For each damage pattern, acceleration responses caused by 2 additional ambient loads (denoted as L1 and L2 in Table 4) are used as the testing data. The selection of SVR parameters, kernel, and features are the same as in the 2-story example.

	Damage pattern 1				Damage pattern 2			
# of patterns	30		100		30		100	
Ambient load	L1	L2	L1	L2	L1	L2	L1	L2
h (1F)	**2.57**	**2.46**	**1.69**	**1.56**	**2.03**	**2.07**	**1.78**	**1.58**
h (2F)	1.74	1.07	1.32	0.88	1.48	1.11	1.28	1.09
h (3F)	**1.30**	**1.43**	1.26	1.07	**2.19**	**1.92**	**1.71**	**1.48**
h (4F)	**1.30**	**1.23**	1.02	0.89	**1.20**	1.11	1.08	1.12

Table 6.4. Damage detection and localization results for damage pattern I and II

Comparing to the results given in [31] and [43], the proposed approach indicate the occurrence and the location of the damage in both damage patterns successfully, whereas the second floor in damage pattern 2 is misclassified as damaged in [31].

SVM has achieved remarkable success in pattern recognition and machine learning, and its continuous development also shed new light on applications in civil engineering. This chapter has described two approaches that applies SVM algorithms to vibration-based damage detection in unsupervised manner, in addition to the supervised SVM introduced earlier by other researchers. By applying novelty detection and regression based learning to vibration-based damage detection, the proposed approach eliminates the need of using data from damaged structure for training. A domain specific kernel is also used that combines structural dynamics and SVM algorithm and produces a more robust statistical model for damage detection. Numerical examples have shown that the single-class approach is capable of detecting the occurrence of damage, and the SVR approach can further indicate the location of the damage accurately.

7. Virtual Store Model

A high level overview of retail store modeling using proposed learning approach is given in this chapter. Here the focus is on the business impacts of using a proximate learning model to simulate a theoretically reasonable virtual store, and avoid the need of doing expensive filed tests when a new technology such as RFID is introduced into the supply chain. Detailed modeling approach can be found in.[44]

7.1 Introduction on Lowering Out of Stocks

The essence of capitalism, the market dynamics of supply and demand, fails miserably when retail customers are faced with out of stocks. This hurts consumers that can not buy the product they want, manufacturers that often see sales going to competitive products, and retailers that see their turnover drop. Indeed, Out-of-stocks (OOS) have been a long-standing and vexing problem within the retail Consumer Package Goods (CPG) industry. Estimates of the percentage of products out of stock vary widely among retailers, but the majority of estimates fall in the range of five to ten percent. The percentage of OOS is even higher for fast moving goods and promotional items, averaging between 15 and 20 percent. OOS levels increase even more on high-traffic days. In contrast, the vision of perfect retail is that product is available precisely when consumers want to purchase it. Despite many new value chain initiatives, recent research suggests that the problem of OOS has not improved significantly.

Due to lack of information on OOS and a scarcity of studies, certain limitations exist with current estimates that have not been thoroughly addressed. First, there is no standardized definition of what constitutes an OOS (see Annex for a collection of different measures). The lack of a definition makes it difficult to compare data and statistics. Also, there is minimal scientific data or analysis on OOS. Many retailers do not have any insight into their own OOS levels and rely on general industry statistics. Much data has been collected via surveys rather than from consumer actions. The future challenge is to transform OOS identification and resolution from an art into a science. This chapter will shed light on OOS and methods to improve OOS using RFID based on the work with Wal-Mart.

Among the many types of OOS one stands out as particularly nagging: Often OOS products on the shelves are present in the backroom of the store. In practice, possibly given the 100,000+ SKUs in a typical Wal-Mart store, it is difficult for store associates to know if a product is in the backroom or not, given that Wal-Mart's current perpetual inventory (PI) system tracks only the total amount of the product. If store associates knew there are products in the backroom when the products are running low on the shelf, they could move such products to the shelf. If the backroom of a store has RFID readers, one can keep track of whether products are in the store backroom or not. It follows naturally then that RFID/EPC could help lower OOS by alerting store associates of which OOS products are in the backroom. In fact, RFID/EPC technology acts as a microscope in the supply chain helping identify where products are at all times. We have found that lack of visibility is the root cause of many of the short-term improvements RFID can bring to the OOS challenge.

Wal-Mart's recent introduction of RFID aimed to lower OOS. To help store associates locate products that were OOS in the sales floor but available in the backroom, Wal-Mart used RFID to identify such products and placed them automatically on the store associate's pick-list for that day. The business logic is that by doing so, store associates will be more effective (from the point of view of lowering OOS) at locating those products and moving them to the sales floor. The goal was to understand qualitatively if the introduction of RFID could provide any advantages and quantitatively to measure the extent of such improvements.

To understand whether Wal-Mart's initial RFID implementation did in fact lower manufacturer's OOS in their stores, we set up a research experiment to validate these hypotheses based on two efforts. The first effort was directed at understanding the processes that impacted OOS and it was based on an analysis of the replenishment process for that manufacturer; we reviewed the processes at the manufacturer's distribution center (DC), at the store's DC and at the store itself. I will discuss the potential impacts on store processes in section 4, as the manufacturer's DC work has been reported elsewhere. The second effort was directed at estimating RFID's quantitative impact on reducing store OOS. The quantitative results I will present are based on the largest scaled out-of-stock field study at Wal-Mart to date, a 29-week OOS report, generated by manually scanning OOS in 12 pilot stores (that had introduced RFID-enabled processes) and 12 control stores everyday. Detailed description of the OOS field study setting and a preliminary trend analysis of the entire observation have been reported by the RFID Research Center at University of Arkansas [45].

The rest of the chapter is organized as follows. In Section 7.2 we introduce a generic retail store model and illustrate how store replenishment is performed in theoretical scenarios. We will see that RFID, or any other enabling technology, will have only marginal contribution in these scenarios. In the real world, however, every single process in the chain could go wrong and lead to delay, OOS, and eventually lost sales. In section 7.3, we will analyze how those assumptions made in the theoretical scenario may break down in the real world, and discuss how RFID can help prevent such break down situations based on store observation work. In section 7.4 we will discuss the quantitative OOS study, and the conclusion will be given in section 7.5.

7.2 Theoretical Scenarios

Wal-Mart runs one of the largest and most complex operations in the world, with sales surpassing the GDP of many countries – yet at the store level, operations are in many ways similar to that of most retailers: the store is divided in departments, and each department has a department head and a series of staff that support it; store associates "walk" the store floor a few times a day looking for OOS (among other tasks) and create a pick-list with items that need to be picked and subsequently "work" the pick-list, i.e., they move pick-list items that are on the backroom to the sales floor; items are re-ordered by the company's information system (IS) based on a number of factors including Point-of-Sales (POS) data, demand forecasting, shelf capacity and on-hand inventory; store associates have the right to adjust the system variables based on their observations (e.g. if there is shrinkage they may reduce the on-hand quantity). A generic model of a barebones retail store is shown in Figure 7.1.

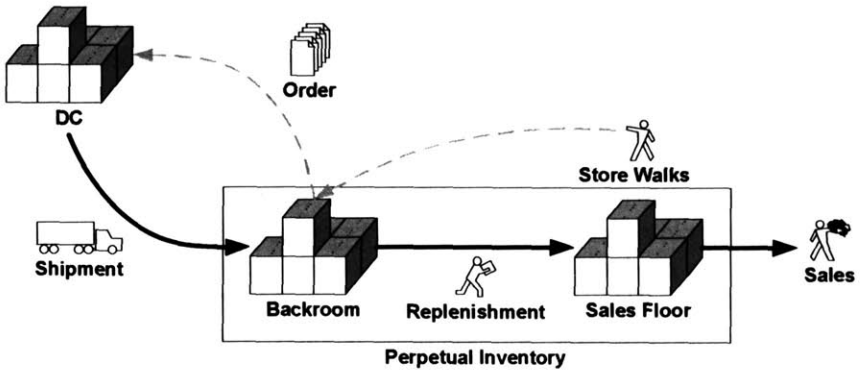


Figure 7.1 Barebones Retail Store Model

In an "ideal" theoretical world (assuming the processes are executed properly and flawlessly), OOS rates should be low. However, as we know, there are many fallacies with a purely theoretical approach that will be highlighted as we search for reasons why execution can deviate from a flawless operation. This will lead us naturally into understanding how RFID can help in improving execution performance.

The performances of the replenishment in the generic retail store model shown in figure 7.1 are determined by seven factors, namely, 1. Accuracy of the perpetual inventory information, 2. Order delay (time required for DC to process an order) and order error, 3. DC in-stock position, 4. Shipment frequency and mis-shipment rate, 5. Accuracy of the demand forecasting, 6. Store walks (zoning) frequency, 7. Store execution rate. To illustrate the logic behind this way of managing inventory, let us construct a theoretical scenario (referred to as "scenario one" hereafter) that would yield no or minimal OOS. Such scenario will be based the following execution assumptions,

Factors	Assumption
Perpetual Inventory Information	No error
Order Delay / Error	No delay / No error
Shipment Frequency / Error	As often as needed / No error
DC in Stock Position	Always in stock
Demand Forecasting	Optimum (cannot be further improved)
Store Walks	Three times a day
Store Execution Rate	100%, i.e., pick-lists are "worked" efficiently

Table 7.1 Assumptions in Theoretical Scenario One

Under those assumptions listed in table 7.1, store associates' judgment combined with system's rules yield optimal forecast, and employees know exactly how much product is in the store and where it is so that orders are made before store run out of products and shelves are replenished before stock-out. New orders and deliveries are constantly coming in, deliveries arrive immediately after being ordered

(imagine that the DC is so close to the store that one could achieve multiple deliveries per day), the DC is not OOS, and there are no replenishment errors across the supply chain. We could also assume that the shelf capacity is large enough to hold purchases of a typical day. However, this last assumption does not add much to the scenario because shelf capacity can be offset by rapid re-ordering and replenishment that is essentially what this scenario assumes.

If this was the case, there would never be any out-of-stocks unless there was a customer (or group of customers) that picked all the products in a given shelf at once – until the item was replenished the shelf would certainly be OOS. We call this the "large party OOS" because it can occur when a customer is purchasing products in bulk, such as when buying "confetti" for a large party. Given these factors, unless there is a "large party OOS" it would be difficult to create an OOS given that store associates would act immediately as stock levels approach OOS. As these factors stated above become further from perfect, OOS grows and forecasting/order-writing skills come into play. Moreover, in reality, only some of these factors can be improved. For example, stores cannot do much to reduce the amount of time a delivery takes since the transit time is necessary and is already relatively short. Before getting into the details of what can and can not go wrong, and how to improve it, let's refine the scenario above to construct a more realistic one.

A more realistic scenario, we will refer to as "scenario two", will help further elucidate some of the possible avenues to addressing OOS. Assume that we are trying to lower a store's OOS for a product category that has the following properties, products can be ordered on every Wednesday and will be delivered on Thursday, i.e., at most one order and delivery per week (deliveries are received multiple times a week in most retail stores and departments). Assume then that there is practically no error in the ordering process nor in the corresponding delivery; human demand forecasting combined with machine forecasting cannot be further improved; the store DC has no OOS; and three store-walks a day. This scenario is essentially scenario one with the two relaxations listed in table 7.2, and it results in OOS on the shelf only when the product is also not in the backroom.

Factors	Assumption
Order Delay / Error	One day / No error
Shipment Frequency / Error	Once per week / No error

Table 7.2 Assumptions Relaxed in Theoretical Scenario Two

In scenario two, observe that OOS will occur and often – however, as we will discuss there is nothing RFID can do to improve this scenario. The discussion of why things can not be improved in this hypothetical scenario will help us understand what can go wrong and what RFID can do to help. Observe that in this scenario, if there is a "large party OOS" on Thursday, the OOS will exist until the next delivery arrives, which could be up to a week. In general, if an OOS occurs and the product is not in the store's backroom, the product will not be replenished until the next delivery cycle. Hence, a major source of OOS in this scenario is unexpected large volume demand. Note that we assume the demand forecasting are optimal does not mean it is perfect, it is just that the forecast cannot be further improved given the information and technology we have. Unless technology helps improve demand forecasting, OOS will not be lowered in this scenario.

In Wal-Mart, as in most retail stores, demand forecasting is not always left to humans, for example, sometimes it is based on POS data, sometimes on DC availability, and sometimes it is based on manufacturer-dictated promotional schedules. Observing that in this scenario, it is a bit irrelevant who initially places the orders, either the system through POS data, through RFID reads, or simply through human intervention. As long as humans are better off at predicting demand, make no errors in ordering, and can overwrite any system decision, all we will do with technology is improve operational costs – not demand forecasting.

We have found that the operations of scenario two are in many ways Wal-Mart's ideal store operations. Management focus is centered on operational effectiveness and on lowering the pain points where translation to reality deviates from this "perfect" world. This is the subject of the next section.

7.3 How RFID Can Help in Principle

We contend that lowering OOS in the scenario 2 is difficult because orders can only be made one day a week and there is no error to correct and demand forecasting accuracy is already optimal. Furthermore, in this scenario OOS are kept at minimum levels because employees are constantly walking the store and replenishing products with back stock. Adding employees to this scenario would simply increase the speed at which large party OOS are handled provided the product is in the backroom. Therefore, there is no OOS reduction via RFID unless the stated assumptions in the theoretical scenario are violated, which, interestingly, is usually the case in reality. This is why we find this theoretical scenario a good starting point to understand how to lower OOS. In this section, we will look at the store processes in a finer granularity, and discuss what may go wrong in each process, the results of the failures, and how RFID can help prevent the failure. To do so, we first introduce a systematic approach to study a process from the cause-effect angle, and describe the decomposed store processes based on the observation work performed in two Wal-Mart stores. Then, using the cause-effect approach, we summarize what can go wrong in a retail store when theoretical assumptions are violated and why we think RFID/EPC has the potential of lowering OOS.

A generic system, to its minimum, can be divided into three parts, namely, input, process, and output. A more sophisticated system can collect information through the processes and outputs and give feedback to the inputs, as shown in figure 7.2.

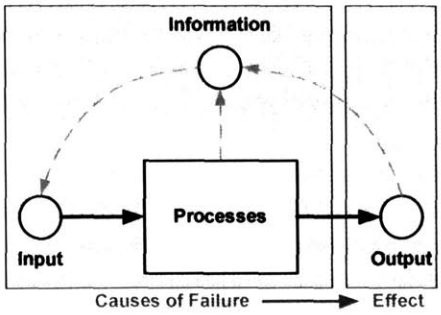


Figure 7.2 A Generic System.

Take a retail store for example, the inputs are the products received in the backroom, the outputs are the sales or other data of interests that can be measured, processes usually includes store walks, creating pick-lists, moving products from backroom to sales floor, ...etc., and large chain stores usually use PI and demand forecasting systems to help manage inventory and place orders automatically, as depicted in figure 3a.

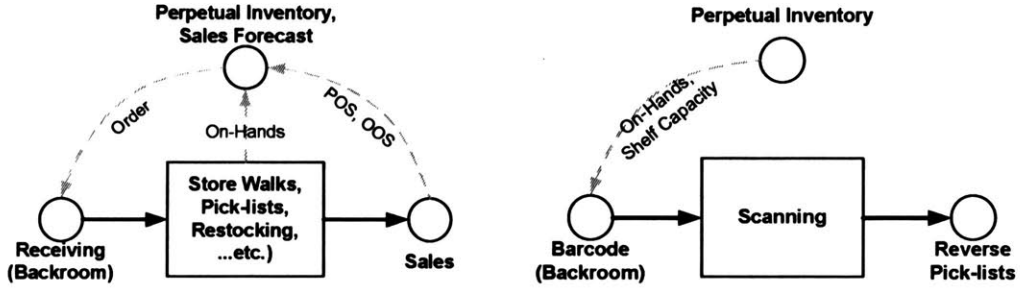


Figure 7.3a Left: A Retail Store Replenishment System.
7.3b Right: A Single Process (Creating Reverse Pick-Lists) in a Retail Store

The methodology can be further extended to a single process in the system, for example, when creating a reverse pick-list, a store associate scans (process) product barcode (input) in the backroom, and if the on-hand quantity is less than the shelf capacity (information), the product will be added to the reverse pick-list (output), as depicted in figure 7.3b. It is now clear that the product of the system, or the result of each process, is highly depended on its input, information and process (ref. figure 7.2). Each of the three causes in a retail store can easily deviate from ideal, for example, wrong products or incorrect quantity of products may be sent to a store and causes input error; human centric in-store processes are prone to error; cashiers may scan the same barcode twice for same price products (e.g. orange juice with or without pulp) to save time, when there are actually two different SKUs and causes error in POS and PI system; and the accuracy of PI is easily affected by theft, shrinkage, sweet-hearting, dusty cases (products exist but cannot be found in the backroom), ...etc., and results in incorrect information.

During the extensive qualitative research, the most significant and likely obvious observation was that the store processes today are highly human-centric. This may not be an issue in a "best practice" store with experienced staff, but can be problematic in average stores, where store processes deviate from the ideal and can benefit from RFID more. The best performing departments or stores tended to be those led by highly experienced staff who managed with a combination of store data / reports, but also significant internal knowledge / experience. Unfortunately, one of the main challenges that the entire retail industry must deal with is the reality of high staff turnover. While RFID/EPC cannot reduce turnover, it can potentially assist in greater consistency across departments / stores and reduce the impact of inevitable turnover.

One goal of the research was to understand what may be the levers that impact OOS and how these may be affected by RFID/EPC enabled processes. To this end, two Wal-Mart Super Stores were selected and extensive field interviews were conducted at both. A researcher spent two months continuously in the stores learning about the processes during the introduction of a major EPC/RFID initiative. In addition to store level observation, and to understand possible impacts across the supply chain, other researchers spent two weeks in a Wal-Mart DC, and three weeks in Gillette's DC to

observe the potential improvements from upstream. Researchers were given full access, neither Wal-Mart nor Gillette put any limitations or objections, and over 200 interviews where conducted to understand the processes. In-store processes are decomposed using the MIT process handbook methodology [46]. Detailed description of each process is beyond the scope of this chapter, and can be found in [47].

During the time of the study, the only major process change at Wal-Mart's RFID pilot stores was the auto-pick process. It is important to understand that RFID/EPC is an enabling technology, and the technology itself will not reduce OOS. However, the process changes enabled by the technology can have great impact on the OOS. Before the auto-pick-list initiative, two types of pick-list generating mechanisms were used at Wal-Mart. A pick-list is a list of products needed to be replenished, and is generated when store associates scan the barcode on the shelf in sales floor, when the product is OOS or running low. The second mechanism, referred to as reverse pick-list, is performed in the backroom. Store associates scan the barcode of products in the backroom, and the system adds the product into pick-list when the total on-hands quantity is less than the maximum shelf capacity. Note that the system does not distinguish the products in the backroom from the products in the sales floor, since no such information is available. On the other hand, with RFID/EPC, products are tracked when moved from backroom, and the system now knows how many products are in the sales floor and how many are left in the backroom. Therefore, as soon as there is enough room on the shelf for more than one pack of product, the system can add the item on auto-pick-list and has the store associates to replenish the shelf, and lift the replenishment process from passive to proactive.

To reduce the impact on store associates and lower the learning curve, auto-pick process uses the same pick-list concept, and simply adds additional products to the regular pick-list. Generating a regular pick-list is a labor intensive process, and auto-pick-list can reduce the time store associates spend on zoning, and give them more time to work the pick-list and help customers. In a typical week in January 2006, a Wal-Mart store had total 2,000 SKUs on its pick-list, where 500 were added by the auto-pick process. However, only 5% of all SKUs in that store were tagged with EPC tag, i.e., only 5% SKUs could have been picked by the auto-pick process. In other words, when all products are tagged, auto-pick process will have the potential to generate 10,000 SKUs on the list. Further, auto-pick process can prioritize the replenishment order based on various factors, such as on-shelf stock level, price, and demand forecast. For instance, if two products have the same shelf capacity and both have 20% on-shelf stock left, but POS data shows that one products higher sales rate would likely result in a sooner OOS, auto-pick-list could conceivably assign a higher priority to that product.

7.4 Quantitative Analysis on OOS Reduction

From decomposing current in-store processes to examining the auto-pick process, the qualitative store observations all supports the notion that RFID/EPC can help lower OOS. In this section, we will present a quantitative analysis of the auto-pick-list initiative to further strengthen the argument. In the quantitative experiment, the OOS level in two groups of retail stores are compared, one group with the auto-pick-lists initiative and one without. As previously mentioned, auto-pick process is expected to generate pick-lists proactively and automatically, and therefore reduce OOS while

reducing the work load from store associates at the same time. The result we have fully supports the store observations.

A group of test stores were chosen at random from among the 104 RFID-enabled stores at the time. In total, 12 pilot stores were selected: 6 Super-centers, 3 Neighborhood Markets, and 3 Division I (i.e., traditional Wal-Mart stores). All stores were in the Dallas, Texas area given that this is the region where the Wal-Mart DCs and stores had been EPC/RFID enabled at the time of the research. Twelve matching control stores were then chosen based on geographic location, size of stores (square footage), and annual sales. These stores (6 Super-centers, 3 Neighborhood Markets, and 3 Division I stores) were located in Texas and southern Oklahoma. 5 stores did not make the final analysis due to the DC – store realignment, and EPC tagged products could no longer be sent to those 5 stores. Although 4 more stores are later added, there was no base period data available in those 4 stores, and thus only 7 pilot stores are used in the following analysis.

From February 14, 2005 to September 12, 2005 (29 full weeks), the test and control stores were scanned daily. A national merchandising group was contracted to perform the scanning. An OOS was defined as any empty shelf space in the study. Almost all sections of the stores were scanned for out of stocks, with some exceptions such as bakery goods, variable weight produce and meat and fresh flowers. The daily scanning of a particular store started at approximately the same time each day and the scanners followed the same route each day. Thus, the same areas were scanned at approximately the same time each day in each store. All items out of stock were scanned regardless of being tagged or not. Initially only 10 items per category could be added to a pick-list by the auto-pick process in the transition period, referred to as the "partial" auto-pick-list period. Later, 100 items per category could be added to the auto-pick list after 2 months, and will be referred to as the "full" auto-pick-list period hereafter.

At the beginning of the study, 4554 unique products, from almost all departments across the various Wal-Mart store formats, contained RFID tags, and this set of products was tracked for OOS across both the pilot and control stores throughout the duration of the study. The study is focused on Gillette products, a subset of the full set. In control stores and the base period of pilot stores, a pick-list was generated as a result of "zoning" activities performed by store associates in sales floors and backrooms. In addition, in the partial and full auto-pick-list period of pilot stores, items were also added to the pick-lists by the system automatically based on the EPC reads. The actual dates of each period, and the corresponding pick-list generation mechanism in each group of stores are illustrated in figure 7.4.

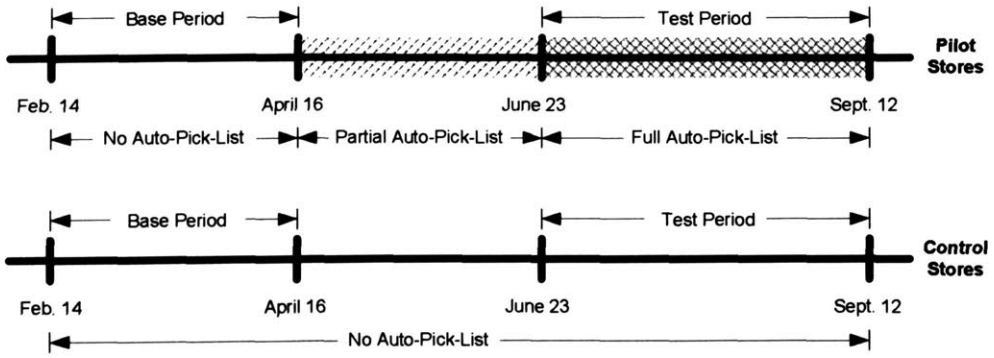


Figure 7.4 Base and Test Periods in Pilot and Control Stores

The major assumption of this study is that the way pick-lists are generated in pilot stores is the only major difference between the base period and the test period. This assumption is regarded reasonable based on the three month store observation before and during the test period. We also assumed the base data was not biased towards EPC/RFID given that those capturing it had no knowledge of which products where EPC/RFID enabled and which ones where not. Strictly speaking, auto-pick is not a process change since it simply adds more items on the existing pick-list, and thus the impact on store execution should be low.

The objective is to study whether there is a statistically significant OOS reduction in the pilot stores after the full auto-pick-list is activated, comparing to the OOS during the base period in the same store. By comparing two different periods within the same store, we can safely eliminate many factors that may affect the OOS, such as store size, store location, store execution, demand variability... etc. A standard hypothesis test procedure was used to examine the impact of auto-pick-list on OOS for Gillette products.

There were 683 different Gillette items (the same products in different sizes or packages were considered as different items) sold in those Wal-Mart stores during the study, 608 of them had EPC tags attached to the case. Note: during the time of study, most products were not tagged to the inner-pack or item level. Hence, items not replenished in full case did not have EPC reads when moved from backroom to sales floor, and would not be picked by auto-pick process. 227 of the 608 tagged items were replenished in full case, and 82 items had been picked by auto-pick-list at lease once. The result of using the 227-item subset is shown in table 7.5, The analysis was complemented by looking at different measures of the OOS data, for example, the 82-item subset, the 227-item subset where the OOS is weighted by the item value, and the 608-item subset. All analyses yielded results very close to Table 7.5. This similarity could be due to the improvements in EPC tagged products helping execution in non-tagged products, so for example, auto-pick process freeing store associates to devote more time work the pick-list, which helped non-tagged products as well as tagged products.

	Worse	Stationary	Better
Pilot	0%	57%	43%
Control	42%	33%	25%

Table 7.5 OOS Comparison between Pilot Stores and Control Stores

Table 7.5 shows that in the control group, where no EPC/RFID related initiative was taken during the time of study, 42% of the stores became worse, 33% had no significant change, and 25% became better in the test period. Here "Worse" means, within 95% confidence interval, there is a statistical significant increase in OOS during the test period in the store, comparing to its OOS in the base period. Similarly, "Better" means there is a significant reduction in OOS. "Stationary" means the change in OOS, either increasing or decreasing, is not significant. The performance in controls stores represented the average store performance across Wal-Mart during the period of study, and stores in the pilot group should have similar distribution, if there were no process changes applied. On the other hand, none of stores in the pilot group became worse, and more stores performed significantly

better in the test period. Since the only major difference in these two groups was the auto-pick process, the results fully supports the hypothesis that the auto-pick process help lowering OOS.

Although the data came from Wal-Mart's largest OOS field study ever conducted, one may argue that 7 pilot stores is not enough to rule out the chance that we had just been "lucky" in selecting those stores. However, what would be the chances of us not seeing any Pilot store that performed worse? If there were one store, in the study, the chances would be 2/3 that we do not pick a worse store (assuming the underlying performance is uniformly distributed, i.e., 1/3 worse, 1/3 equal, 1/3 better). The probability of no worse store in all seven pilot stores would be $(2/3)^7$, i.e. less than 6%.If we use the performance in control stores as the base, the probability is then $0.58^7 = 2\%$. In other words, if RFID had no impact, only in 2% of the cases we would have a result similar to the one we have obtained, which supports the believe that RFID/EPC is helping the last 100 yards of the supply chain.

$$OOS_{\%} = \beta_0 + \beta_1\left(\frac{\sigma_{POS}}{\mu_{POS}}\right) + \beta_2\left(\frac{SC_{Max}}{\mu_{POS}}\right) + \beta_3\left(b_{RFID}\right) + \varepsilon$$

Percentage of OOS which can be reduced by auto-pick-list $= \beta_0 + \beta_1$ Demand Variability $+ \beta_2$ Shelf Capacity $+ \beta_3$ RFID and Everything not modelled $+$ Error

Figure 7.5 Domain Knowledge in OOS Modeling and the Linearized Model

Furthermore, by combing SVM with the domain knowledge shown in figure 7.5, instead of the original linear model used by other researches, a more accurate model and OOS prediction can be obtained. The linear model has an R-square equal to 0.12, and the SVM model boosted the value to 0.57, while keeps a much smaller generalization error.

7.5 Business Impact

OOS is a challenging metric because the problem is the flip side of the problem of minimizing inventory. OOS could be solved by carrying a lot of inventory; however, it increased the inventory costs and is not desirable. The ideal solution would be to find the optimal point where OOS is low and the operation costs are minimized. To this end, Wal-Mart introduced EPC/RFID, which provides a whole new level of granularity into its supply chain, and sheds lights on new approaches of lowering OOS.

As stated before, EPC/RFID have only marginal benefit to a best practice store, where store associates are experienced and store processes are performed flawlessly. However, based on the store observation work, we discovered that even in a data-centric retailer like Wal-Mart, the processes are still highly human-centric in the store end. By decomposing the replenishment process, we discussed what can go wrong in today's human-centric retail store, the impact of failure on OOS, and why we think RFID can help in each process. Specifically, we believe that EPC/RFID could not only improve the overall effectiveness of store-employees, but also the consistency of performance across departments and stores, as more "actionable" recommendations are automatically developed with the assistance of EPC visibility and other business data.

In the quantitative experiment, we analyzed data collected through Wal-Mart's largest OOS field study. The result shows that all RFID (pilot) stores had either significant improvement or stationary behavior during the time of study, while 42% of the non-RFID (control) stores performed significantly worse at the same time. Although the actual amount of OOS events collected by the 29-week long, labor-intensive manual process could be inaccurate, the result still stands as long as the measurement is done consistently across the time of study, since we are comparing the performance of the same store during the two periods. Further, since it is easier to verify an OOS position than to find one, data errors would most likely be false negatives suggesting the OOS opportunity may be bigger than reported. It is important to note that the RFID technology does not affect OOS, if there is no related process change. During the period of study, the only related initiative implemented in pilot stores is the auto-pick process, and the quantitative study supports the assumption that the RFID enabled auto-pick process is more efficient than the regular pick-list process.

All the above is consistent with an incipient and promising RFID deployment at Wal-Mart stores. Given that the store observation work suggests many possible process improvements enabled by RFID, we are firmly convinced that the auto-pick-list is juts the beginning of a journey to improve customer satisfaction.

8. Conclusion

8.1 Summary

A generic statistical learning framework that supports data mining, modeling, and simulation of dynamic systems is presented. The proposed algorithm takes both domain knowledge and data measured from the system into consideration and provide a mechanism that combines domain knowledge and statistical learning. An efficient approximation solver is suggested that allow the algorithm to solve large-scale problem efficiently. The framework is built on a serviced-oriented architecture that provides a loosely-coupled, robust, and highly interoperable software system. Also, several benchmark problems and a practical engineering are used to validate the methodology.

Unprocessed raw data sets are not very useful per se. To support decision making, what we need is the information and knowledge derived from the data. For most nontrivial problems, the size of data set makes analytical analysis infeasible. Probability based algorithms draw from several different yet closely related disciplines, such as machine learning, data mining, statistical analysis, and pattern recognition, are commonly used when dealing with large-scale problems. Depending on how an algorithm models the underlying probability distribution of a system, it can be separated into one of the two groups, discriminative or generative model. Generally speaking, discriminative approaches are leading in both performance and accuracy when data are abundant. In particular, support vector machine, one of the latest and most successful statistical learning approaches, is used as the foundation of the work. However, many challenges exist when we are solving practical, large-scale system problems with learning algorithms. First, discriminative approach is more sensitive to data quality because of the lack of prior knowledge. Incorporating prior knowledge into discriminative learning is an active research area, and a systematic approach of transforming domain knowledge of systems into learning models is still lacking. Second, learning is a computational extensive process and often become intractable when the sample size exceeds several thousands. Third, the acceptance and adoption of machine learning has not been established in most engineering and business fields. Successful applications are not enough and learning algorithms are often applied as is and not tuned to the problem. Last but not least, a robust learning framework that can be customized and embedded into existing systems is still missing.

The resented work itself is constraint optimization problem. The goal is to find a best quality model that is a robust and realistic representation the underlying system, subject to the limited resources. If we have unlimited resources, for example but not limited to, man power, money, and time, we can conduct field tests directly. When we do not, mathematical modeling and simulation become the method of choice. If we have enough knowledge and domain experts, we can build a realistic high-order model that closely represents the system of interest. When we do not, a probability-based proximate model is a reasonable alternative. If we have enough computational resources, we can solve the search and optimization problems embedded in statistical models with brute force and there is do not need kernels. With a kernel, we no longer have information about each individual observation, what we have is a Gram matrix that summarizes the relation of each observation to all others. A different choice of kernel corresponds to a different metric measuring the relation. The kernel then becomes an information bottleneck that imposes unrecoverable information loss. Still,

large-scale problem impose severe computation challenges to kernel based algorithms, especially in the presented work that domain knowledge is incorporated through explicitly constructed kernels.

8.2 Contributions

A few algorithms have been proposed to bridge the gap between discriminative and generative learning, and later their statistical properties have also been studied. However, several difficulties are still remained. First, there is no systematic approach of transforming a dynamic system into a probabilistic model. Second, a generative learning scheme is often represented as a directed acyclic graph, which can not capture the feedback relation commonly required in a real world system. Third, a flexible and robust framework to support discriminative learning in generative models is missing. Last but not least, when performing simulation, a model is no longer stationary and a system with orchestration mechanisms is also required to organize the work flow.

The objective of this work is two-fold. On the business side, we want to provide a flexible virtual system that can conduct experiments that are expensive or impossible to perform field tests, help evaluating the value of new process changes in the system, and optimize the system through a measure-model-control loop. On the technological side, we want to deliver a hybrid model utilizes the best of algorithms drawn from system dynamics and machine learning and develop a robust system to incorporate learning with modeling and simulation. In this work, we present such a system, along with the algorithms, design issues, and experiments of large scale, real world problems.

The first contribution is to develop a general purposed kernel that can be used to introduce domain knowledge into support vector machine learning. The second contribution is to provide an efficient solver that can handle very large data sets. It takes advantages of the sparseness of support vectors and can be parallelized easily to further speed-up the computation. The third contribution is to provide a robust, flexible, yet lightweight implementation of the proposed framework. It is written in a platform independent language and can be integrated into existing systems or embedded into databases. Further, the implementation is compatible with modern web protocols, and can be exposed as a web-service, or as the intelligent core of a larger service. The fourth contribution is to apply the proposed framework on practical engineering and business problems, showing that data, together with domain knowledge and a customizable learning framework, is well suited for solving various system analysis problems.

Bibliography

1. Jansen, J.D., et al., *Closed-loop Reservoir Management.* First Break, 2005. **23**: p. 43-48.
2. Williams, J.R. and C.-H. Tsou, *Production Optimization through Modeling, Simulation and Discriminative Learning*. 2005, IESL, Massachusetts Institute of Technology: Cambridge.
3. Whitcomb, G. and C. Gallagher, *Wal-Mart Begins Roll-Out Of Electronic Product Codes in Dallas/Fort Worth Area*, in *Wal-Mart Press Releases*. 2004.
4. Mitchell, T.M., *Does Machine Learning Really Work?* in *AI Magazine*. 1997. p. 11-20.
5. Schapire, R.E. *A Brief Introduction to Boosting*. in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. 1999.
6. Mitchell, T.M., *Machine Learning*. 1997: McGraw-Hill.
7. Hadamard, J., *Sur les problèmes aux dérivées partielles et leur signification physique*. 1902, Princeton University Bulletin. p. 49-52.
8. Vapnik, V.N., *Statistical Learning Theory*. Adaptive and Learning Systems for Signal Processing, Communications, and Control, ed. S. Haykin. 1998: Wiley-Interscience.
9. Vapnik, V.N. and A.Y. Chervonenkis, *Theory of Pattern Recognition.* Moscow: Nauka. (In Russian), 1974.
10. Byvatov, E., et al., *Comparison of support vector machine and artificial neural network systems for drug/nondrug classification.* Journal of Chemical Information and Computer Sciences, 2003. **43**(6): p. 1882-1889.
11. Slater, M.L., *A Note on Motzkin's Transposition Theorem.* Econometrica, 1951. **19**(2): p. 185-187.
12. Schölkopf, B. and A.J. Smola, *Learning with Kernels - Support Vector Machines, Regularization, Optimization and Beyond.* 2002: MIT Press.
13. Kuhn, H.W. and A.W. Tucker, *Nonlinear Programming.* Proceedings of 2nd Berkeley Symposium, 1951: p. 481-492.
14. Boser, B.E., I. Guyon, and V. Vapnik, *A Training Algorithm for Optimal Margin Classifiers.* Computational Learing Theory, 1992: p. 144-152.
15. Micchelli, C.A., *Algebraic aspects of interpolation.* Proceedings of Symposia in Applied Mathematics, 1986. **36**: p. 81-102.
16. Smola, A.J. and B. Schölkopf, *A Tutorial on Support Vector Regression*, in *NeuroCOLT2 Technical Report*. 1998, Royal Holloway College, University of London, UK.
17. Jaakkola, T., M. Meila, and T. Jebara, *Maximum Entropy Discrimination.* Advances in Neural Information Processing Systems, 1999. **12**.
18. Hoeffding, W., *Probability Inequalities for Sums of Bounded Random Variables.* Journal of the American Statistical Association, 1963. **58**(301): p. 13–30.
19. Burges, C.J.C., *A Tutorial on Support Vector Machines for Pattern Recognition.* Knowledge Discovery and Data Mining, 2(2), 1998.
20. Lustig, I., R. Marsten, and D. Shanno, *On Implementing Mehrotra's Predictorcorrector Interior Point Method for Linear Programming.* SIAM Journal on Optimization, 1992. **2**: p. 435-449.

21. Vapnik, V.N., *Estimation of Dependences Based on Empirical Data [in Russian].* Nauka, 1979 (English translation Springer Verlag, 1982).

22. Osuna, E., R. Freund, and F. Girosi. *An Improved Training Algorithm for Support Vector Machines.* in *Proceedings of IEEE Neural Network for Signal Processing (NNSP).* 1997.

23. Mark, P., E. Shinichi, and S. Halbert, *Cross-Validation Estimates IMSE.* Advances in Neural Information Processing Systems, 1994.

24. Efron, B., *Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation.* Journal of the American Statistical Association, 1983. **78**: p. 316-331.

25. Rytter, A., *Vibration based inspection of Civil Engineering structures*, in *Department of Building Technology and Structural Engineering.* 1993, University of Aalborg: Denmark.

26. Doebling, S.W., C.R. Farrar, and M.B. Prime, *A Summary Review of Vibration-Based Damage Identification Methods.* The Shock and Vibration Digest, 1998. **30**(2): p. 91-105.

27. Farrar, C.R., S.W. Doebling, and D.A. Nix, *Vibration-Based Structural Damage Identification.* Philosophical Transactions of the Royal Society: Mathematical, Physical & Engineering Sciences, 2001. **359**(1778): p. 131-149.

28. Ansari, F., *Sensing Issues in Civil Structural Health Monitoring.* 2005: Springer.

29. Haritos, N. and J.S. Owen, *The Use of Vibration Data for Damage Detection in Bridges: A Comparison of System Identification and Pattern Recognition Approaches.* International Journal of Structural Health Monitoring, 2004.

30. Sohn, H. and C.R. Farrar, *Damage Diagnosis Using Time Series Analysis of Vibration Signals.* Smart Materials and Structures, 2001(10).

31. Lei, Y., et al. *An Enhanced Statistical Damage Detection Algorithm Using Time Series Analysis.* in *Proceedings of the 4th International Workshop on Structural Health Monitoring.* 2003.

32. Worden, K. and A.J. Lane, *Damage Identification using Support Vector Machines.* Smart Materials and Structures, 2001. **10**(3): p. 540-547.

33. Yun, C.B., et al., *Damage Estimation Method Using Committee of Neural Networks.* Smart Nondestructive Evaluation and Health Monitoring of Structural and Biological Systems II. Proceedings of the SPIE, 2003. **5047**: p. 263-274.

34. Barai, S.V. and P.C. Pandey, *Vibration Signature Analysis Using Artificial Neural Networks.* Journal of Computing in Civil Emgineering, 1995: p. 259-265.

35. Bulut, A., P. Shin, and L. Yan. *Real-time Nondestructive Structural Health Monitoring using Support Vector Machines and Wavelets.* in *Proceedings of Knowledge Discovery in Data and Data Mining.* 2004. Seattle, WA.

36. Fugate, M.L., H. Sohn, and C.R. Farrar. *Unsupervised Learning Methods for Vibration-Based Damage Detection.* in *Proceedings of the 18th International Modal Analysis Conference.* 2000. San Antonio, Texas.

37. Schölkopf, B., et al., *Estimating the Support of a High-Dimensional Distribution.* Neural Computation, 2001. **13**: p. 1443-1471.

38. Mead, W.C., et al. *Prediction of Chaotic Time Series using CNLS-Net-Example: The Mackey-Glass Equation.* in *Nonlinear Modeling and Forecasting.* 1992: Addison Wesley.

39. Johnson, E.A., et al. *A Benchmark Problem for Structural Health Monitoring and Damage Detection.* in *Proceedings of the 14th Engineering Mechanics Conference.* 2000. Austin, Texas.

40. Chapelle, O., et al., *Choosing Multiple Parameters for Support Vector Machines.* Machine Learning, 2002. **46**: p. 131-159.

41. Cherkassky, V. and Y. Ma, *Practical selection of SVM parameters and noise estimation for SVM regression.* Neural Networks, 2004. **17**(1): p. 113-126.

42. Fan, R.-E., P.-H. Chen, and C.-J. Lin., *Working set selection using second order information for training SVM.* Journal of Machine Learning Research, 2005. **6**: p. 1889-1918.

43. K.K. Nair, et al. *Application of time series analysis in structural damage evaluation.* in *Proceedings of the International Conference on Structural Health Monitoring.* 2003. Tokyo, Japan.

44. Tsou, C.-H., B. Subirana, and S. Sarma, *Evaluating the Impact of Proactive Pick-list on Out-of-Stock in Retail Stores.* 2007, MIT Auto-ID Laboratory.

45. Hardgrave, B.C., M. Waller, and R. Miller, *Does RFID Reduce Out of Stock? A Preliminary Analysis.* 2005, Information Systems Department, Sam M. Walton Colledge of Business, University of Arkansas.

46. Malone, T.W., K. Crowston, and G.A. Herman, *Organizing Business Knowledge: The MIT Process Handbook.* 2003: The MIT Press.

47. *Wal-Mart In-Store Processes: A Study of Existing Processes in Wal-Mart Super Centers.* 2005, MIT Auto-ID Lab: Cambridge.